CoreSight™ SoC

Revision: r2p1

Technical Reference Manual

ARM®

ARM DDI 0480D (ID010213)

CoreSight SoC
Technical Reference Manual

Release Information

Change history

Date	Issue	Confidentiality	Change
04 November 2011	A	Non-Confidential	First release for r0p0.
16 April 2012	B	Non-Confidential	First release for r1p0.
27 September 2012	C	Non-Confidential	First release for r2p0.
14 December 2012	D	Non-Confidential	First release for r2p1.

Proprietary Notice

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Product Status

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

Web Address

`http://www.arm.com`

Contents
CoreSight SoC Technical Reference Manual

Preface

This preface introduces the *CoreSight™ SoC Technical Reference Manual*. It contains the following sections:

About this book

This is the *Technical Reference Manual* (TRM) for the CoreSight SoC components.

Product revision status

The r*n*p*n* identifier indicates the revision status of the products 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 the following audiences:

- Hardware and software engineers who want to incorporate CoreSight SoC into their design and produce real-time instruction and data trace information from an ASIC.
- Software engineers writing tools to use CoreSight SoC.

This book assumes that readers are familiar with AMBA bus design and JTAG methodology.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this for an overview of the CoreSight components.

Chapter 2 *Functional Overview*

Read this for a description of the major functional blocks and the operation of CoreSight SoC.

Chapter 3 *Programmers Model*

Read this for a description of the memory map and registers.

Chapter 4 *Debug Access Port*

Read this for a description of the *Debug Access Port* (DAP) components.

Chapter 5 *ATB Interconnect Components*

Read this for a description of the *Advanced Trace Bus* (ATB) interconnect components.

Chapter 6 *Timestamp Components*

Read this for a description of the timestamp components.

Chapter 7 *Embedded Cross Trigger*

Read this for a description of the *Embedded Cross Trigger* (ECT) components.

Chapter 8 *Trace Port Interface Unit*

Read this for a description of the *Trace Port Interface Unit* (TPIU) components.

Chapter 9 *Embedded Trace Buffer*

Read this for a description of the *Embedded Trace Buffer* (ETB) components.

Chapter 10 *Granular Power Requestor*

Read this for a description of the *Granular Power Requester* (GPR) components.

Appendix A *Signal Descriptions*

Read this for a description of the CoreSight SoC component signals.

Appendix B *Revisions*

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

Glossary

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

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

Conventions

Conventions that this book can use are described in:

- *Typographical conventions*.
- *Timing diagrams*.
- *Signals* on page ix.

Typographical conventions

The following table describes the typographical conventions:

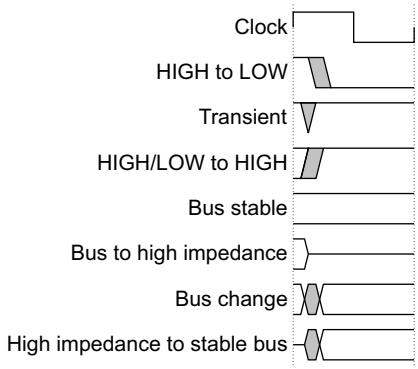
Typographical conventions

Style	Purpose
italic	Introduces special terminology, denotes cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`mono`space	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`monospace` *`italic`*	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the *ARM glossary*. For example, IMPLEMENTATION-DEFINED, IMPLEMENTATION-SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Timing diagrams

The figure named *Key to timing diagram conventions* on page ix 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 and by third parties.

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

ARM publications

This document contains information that is specific to the CoreSight SoC components. See the following documents for other relevant information:

- *CoreSight SoC Implementation Guide* (ARM DII 0267).
- *CoreSight SoC User Guide* (ARM DUI 0563).
- *CoreSight SoC Integration Manual* (ARM DIT 0037).
- *CoreSight Technology System Design Guide* (ARM DGI 0012).
- *Integrating ARM® Cortex™-A Series Processors with CoreSight SoC Integration Manual* (ARM DIT 0044).
- *Integrating ARM Cortex-M Series Processors with CoreSight SoC Integration Manual* (ARM DIT 0049).
- *Integrating ARM Cortex-R Series Processors with CoreSight SoC Integration Manual* (ARM DIT 0048).
- *CoreSight Architecture Specification* (ARM IHI 0029).
- *CoreLink® TrustZone Address Space Controller TZC-380 Technical Reference Manual* (ARM DDI 0431).
- *AMBA™ AHB Trace Macrocell (HTM) Technical Reference Manual* (ARM DDI 0328).

- *Systems IP ARM11 AMBA AHB Extensions* (ARM IHI 0023).
- *AMBA 3 APB Protocol Specification* (ARM IHI 0024).
- *AMBA 4 ATB Protocol Specification* (ARM IHI 0032).
- *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2* (ARM IHI 0031).
- *CoreSight System Trace Macrocell Technical Reference Manual* (ARM DDI 0444).
- *CoreSight Trace Memory Controller Technical Reference Manual* (ARM DDI 0461).

Other publications

This section lists relevant documents published by third parties:

- *IEEE 1149.1-2001 IEEE Standard Test Access Port and Boundary Scan Architecture (JTAG)*.

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 0480D.
- The page numbers to which your comments apply.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Note

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

Chapter 1
Introduction

This chapter introduces CoreSight SoC. It contains the following sections:

1.1 About CoreSight SoC

CoreSight SoC is a set of highly configurable components that you can use to:

- Graphically configure all acceptable configuration options of components using AMBA Designer.
- Build systems that integrate CoreSight SoC components and processors.

  Note

  You can also integrate other CoreSight components like *System Trace Macrocell* (STM) and *Trace Memory Controller* (TMC) that are licensed separately.

- Run the design rule checks to ensure that the configurations match the compliance limitations.
- Create a testbench infrastructure to run system-level tests to verify the debug and trace system operation.

1.1.1 CoreSight SoC features

CoreSight SoC has the following features:

- Configurability.
- Debug and trace visibility of whole systems.
- Cross triggering support between *System-on-Chip* (SoC) subsystems.
- Multi-source trace in a single stream.
- Higher data compression than previous solutions.
- Standard programmers models for tool support.
- Open interfaces for third-party IP.
- Low pin count.
- Low silicon overhead.

1.1.2 Structure of CoreSight SoC

The CoreSight SoC components are grouped into the following categories:

Control and access components

Provide access to other debug components and control of debug behavior. Examples include:

- DAP. See Chapter 4 *Debug Access Port*.
- The *Embedded Cross Trigger* (ECT). See Chapter 4 *Debug Access Port*.

Sources

Generate trace data for output through the *Advanced Trace Bus* (ATB). Examples include:

- *Program Trace Macrocell* (PTM).
- *System Trace Macrocell* (STM), documented separately. See *Additional reading* on page ix.
- CoreSight *Embedded Trace Macrocells* (ETMs), documented separately. See *Additional reading* on page ix.

Links

Provide connection, triggering, and flow of trace data. Examples include:

- Synchronous 1:1 ATB bridge.

- Replicator.
- Trace funnel.

See Chapter 5 *ATB Interconnect Components*.

Sinks

End points for trace data on the SoC. Examples include:

- *Trace Port Interface Unit* (TPIU) for output of trace data off-chip. See Chapter 8 *Trace Port Interface Unit*.
- ETB for on-chip storage of trace data in RAM. See Chapter 9 *Embedded Trace Buffer*.
- *Serial Wire Output* (SWO) for output of *System Trace Macrocell* (STM) trace through a single pin. See *Additional reading* on page ix.

Timestamp

Generates and transports timestamp across the SoC. Examples include:

- Timestamp generator for generating the timestamp.
- Timestamp encoder.
- Timestamp decoder.

See Chapter 6 *Timestamp Components*.

1.2 CoreSight SoC block summary

Table 1-1 shows the CoreSight SoC blocks and their current versions.

Table 1-1 CoreSight block summary

Block name	Description	Block version	Revision in programmers model
CXCTI	Cross trigger interface	r0p4	4
CXCTM	Cross trigger matrix	r0p3	-
CXETB	Embedded trace buffer	r0p3	3
CXATBREPLICATOR	ATB replicator	r1p0	1
CXAUTHREPLICATOR	Authentication replicator	r0p0	-
CXEVENTASYNCBRIDGE	Event asynchronous bridge	r0p0	-
CXTSREPLICATOR	Timestamp replicator	r0p0	-
CXATBFUNNEL	Trace funnel	r1p0	2
CXTPIU	Trace port interface unit	r0p4	4
CXATBUPSIZER	ATB upsizer	r0p0	-
CXATBDOWNSIZER	ATB downsizer	r0p0	-
CXATBASYNCBRIDGE	ATB asynchronous bridge	r0p0	-
CXATBSYNCBRIDGE	ATB synchronous bridge	r0p0	-
CXAPBASYNCBRIDGE	APB asynchronous bridge	r0p0	-
CXAPBSYNCBRIDGE	APB synchronous bridge	r0p1	-
CXAUTHASYNCBRIDGE	Authentication asynchronous bridge	r0p0	-
CXAUTHSYNCBRIDGE	Authentication synchronous bridge	r0p0	-
CXTSGEN	Timestamp generator	r0p0	0
CXTSE	Timestamp encoder	r0p1	-
CXNTSREPLICATOR	Narrow timestamp replicator	r0p0	-
CXNTSASYNCBRIDGE	Narrow timestamp asynchronous bridge	r0p0	-
CXNTSSYNCBRIDGE	Narrow timestamp synchronous bridge	r0p1	-
CXTSD	Timestamp decoder	r0p0	-
Debug Access Port blocks			
CXDAPAHBAP	AHB access port	r0p6	6
CXDAPAPBAP	APB access port	r0p4	4
CXAPBIC	APB interconnect	r0p1	-
CXDAPJTAGAP	JTAG access port	r0p2	2
CXDAPASYNCBRIDGE	DAPBUS asynchronous bridge	r0p0	-
CXDAPSYNCBRIDGE	DAPBUS synchronous bridge	r0p0	-

Table 1-1 CoreSight block summary (continued)

Block name	Description	Block version	Revision in programmers model
CXDAPAXIAP	AXI access port	r0p2	2
CXDAPBUSIC	DAPBUS interconnect	r0p0	-
CXDAPSWJDP	Serial wire and JTAG debug port:	-	-
	• DAPSWDP.	r1p2	5
	• DAPJTAGDP.	r0p5	5
CXGPR	Granular Power Requestor	r0p0	0
CXTSINTP	Timestamp Interpolator	r0p0	-

Note

The revision field of the identification registers is used to check the block version, for blocks with a programmers model.

1.3 Typical CoreSight debugging environment

Figure 1-1 shows an example of CoreSight components in a debugging environment.

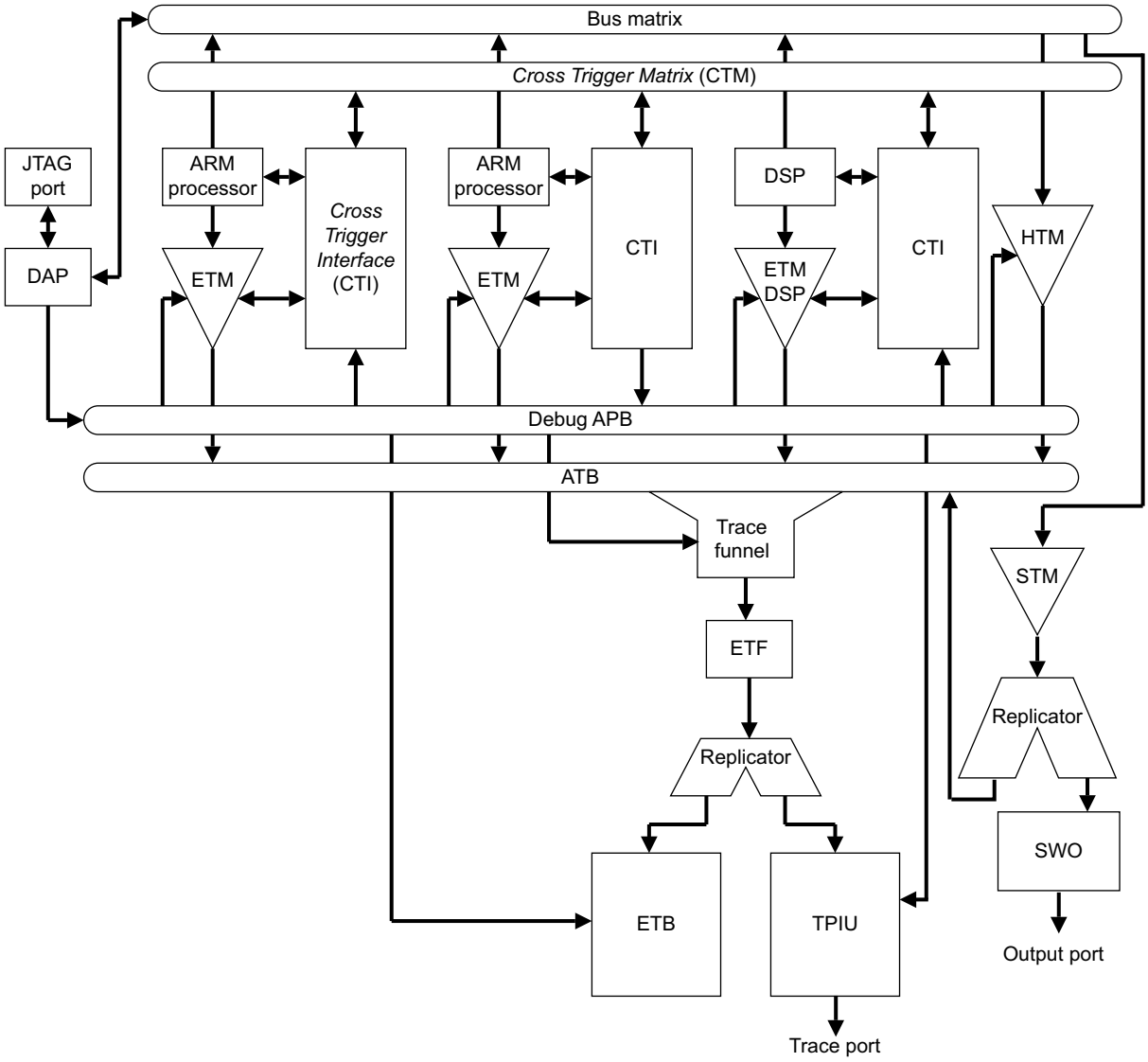


Figure 1-1 CoreSight debugging environment

1.4 Product revisions

This section describes the differences in functionality between product revisions:

r0p0 First release.

r0p0-r1p0 Two new components added:

- Granular Power Requestor. See Chapter 10 *Granular Power Requestor*.
- Timestamp Interpolator. See *Timestamp interpolator* on page 6-12.

r1p0-r2p0 r2p0 includes fixes for all known engineering errata relating to r1p0.

r2p0-r2p1 r2p1 includes fixes for IPXACT changes.

Chapter 2
Functional Overview

This chapter introduces the components available for building a CoreSight SoC trace and debug infrastructure. It describes the basic function of each block and its I/O signals. The configurable parameters that affect block-level signals are shown so that an accurate pinout is reflected for all signals that are external to the block. This is not an exhaustive list or description of all available product customizations. For more information about configurable and programmable features, see *Additional reading* on page ix.

This chapter contains the following sections:

2.1 DAP components

The following components are used to configure a standard interface from a debugger to the rest of debug and trace system:

- *Serial Wire or JTAG - Debug Port*.
- *DAP bus interconnect*.
- *DAP asynchronous bridge* on page 2-3.
- *DAP synchronous bridge* on page 2-3.
- *JTAG - Access Port* on page 2-4.
- *Advanced eXtensible Interface - Access Port* on page 2-4.
- *Advanced High-performance Bus - Access Port* on page 2-5.
- *Advanced Peripheral Bus Access Port* on page 2-6.
- *APB Interconnect with ROM table* on page 2-6.
- *APB asynchronous bridge* on page 2-7.
- *APB synchronous bridge* on page 2-8.

2.1.1 Serial Wire or JTAG - Debug Port

The *Serial Wire or JTAG-Debug Port* (SWJ-DP) connects either a Serial Wire Debug or JTAG probe to the CoreSight SoC debug system. This is the entry point into the debug infrastructure from the external debugger through the *Debug Port* (DP). Figure 2-1 shows the external connections on the SWJ-DP.

See Chapter 4 *Debug Access Port* for more information.

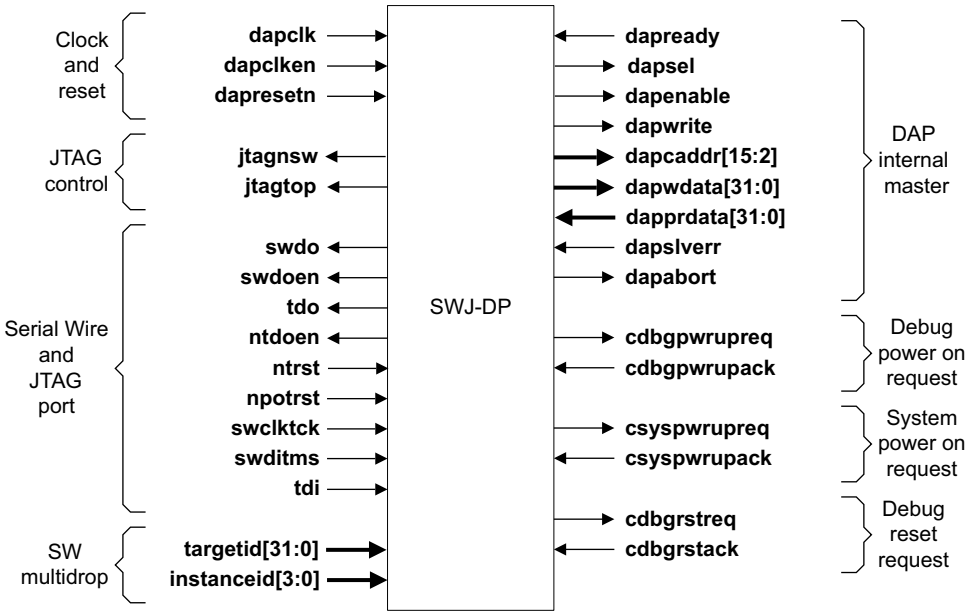


Figure 2-1 SWJ - DP block diagram

2.1.2 DAP bus interconnect

The DAPBUS interconnect connects a debug port to a configurable number of access ports.

The DAPBUS interconnect operates in a single clock domain. The designer must use asynchronous bridges to connect other components that are not synchronous to the DAPBUS interconnect.

The SoC designer must configure the number of master ports within the range 1-32. The required number of master ports is instantiated and suffixed with an interface number.

Figure 2-2 shows the external connections on the DAPBUS interconnect.The interface number of each master port is represented using <*x*>.

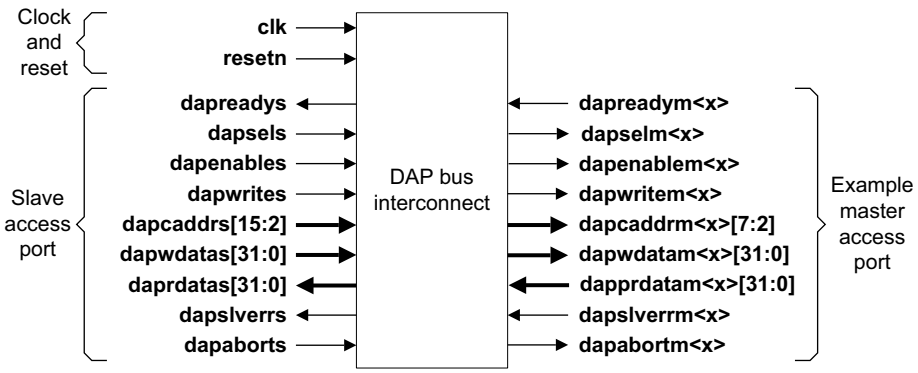


Figure 2-2 DAPBUS interconnect block diagram

2.1.3 DAP asynchronous bridge

The DAP asynchronous bridge enables data transfer between two asynchronous clock domains within the DAP sub-system. Figure 2-3 shows the external connections on the DAP asynchronous bridge.

The AMBA-compliant *Low-power Interface* (LPI) is optional on the DAP asynchronous bridge.

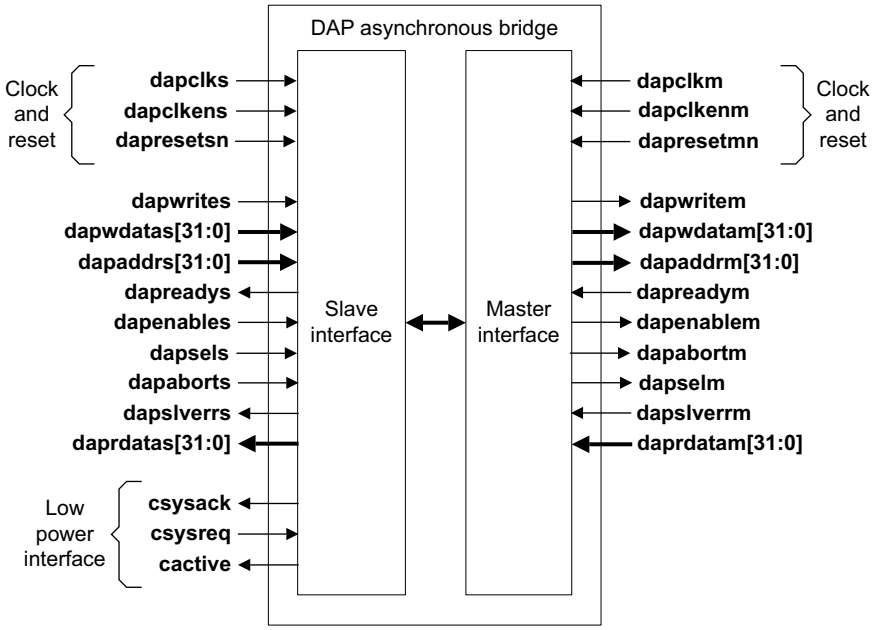


Figure 2-3 DAP asynchronous bridge block diagram

2.1.4 DAP synchronous bridge

The DAP synchronous bridge enables data transfer between two synchronous clock domains within the DAP sub-system. Figure 2-4 on page 2-4 shows the external connections on the DAP synchronous bridge.

The AMBA-compliant LPI is optional on the DAP synchronous bridge.

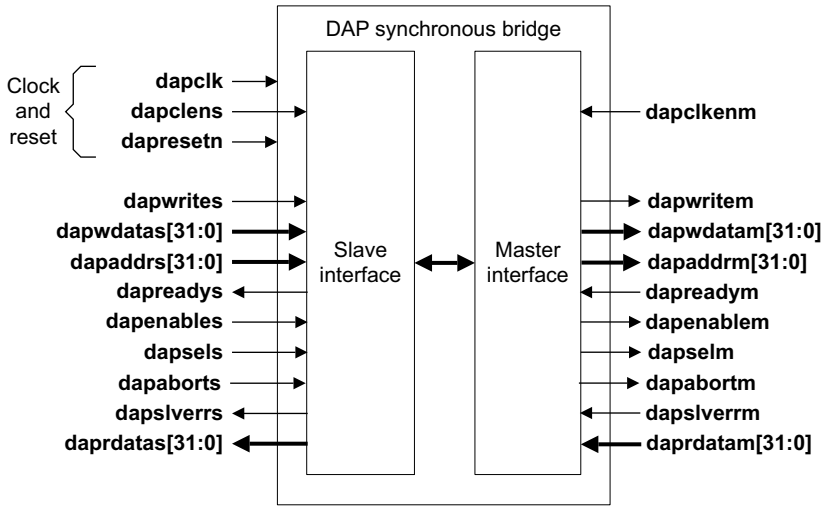


Figure 2-4 DAP synchronous bridge block diagram

2.1.5 JTAG - Access Port

The JTAG-AP provides a dedicated on-chip JTAG access to components from the DAP, independent of the off-chip JTAG or SW connection. See Chapter 4 *Debug Access Port* for more information.

Figure 2-5 shows the external connections on the JTAG-AP.

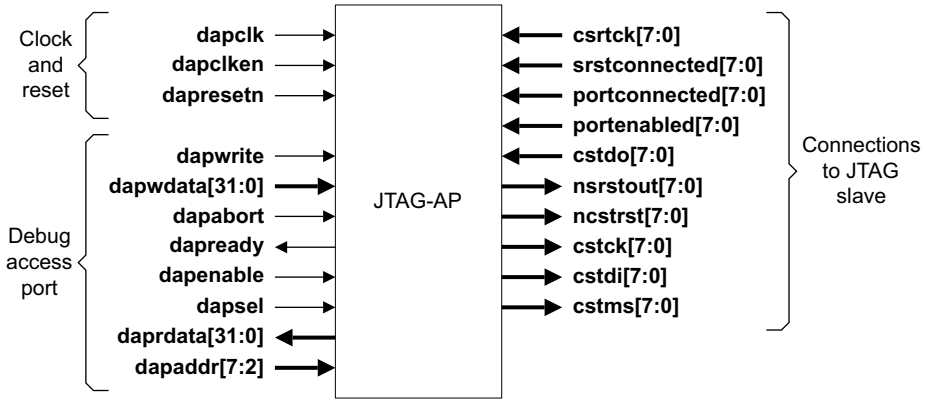


Figure 2-5 JTAG-Access Port block diagram

2.1.6 Advanced eXtensible Interface - Access Port

The *Advanced eXtensible Interface - Access Port* (AXI-AP) connects a DAP to an AXI component or system.

You must configure the AXI-AP during implementation, with the following parameters:

- `AXI_ADDR_WIDTH`, 32-bit or 64-bit. See `aw` in Figure 2-6 on page 2-5.
- `AXI_DATA_WIDTH`, 32-bit or 64-bit. See `dw` in Figure 2-6 on page 2-5.

See Chapter 4 *Debug Access Port* for more information.

Figure 2-6 on page 2-5 shows the external connections on the AXI-AP.

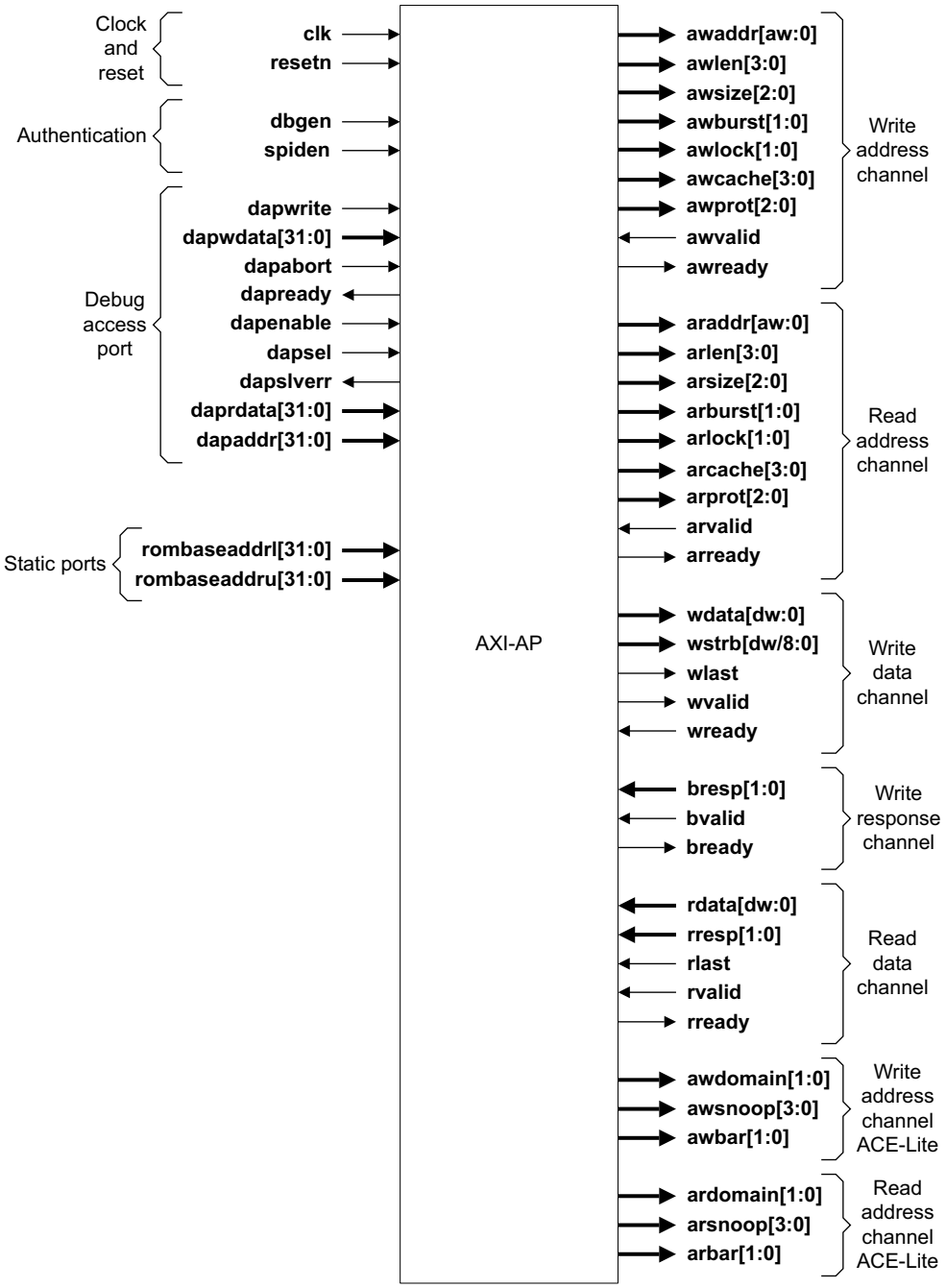


Figure 2-6 Advanced eXtensible Interface - Access Port block diagram

2.1.7 Advanced High-performance Bus - Access Port

The *Advanced High-performance Bus - Access Port* (AHP-AP) connects a DAP to an AHB system bus. The AHB-AP has no configurable features.

See Chapter 4 *Debug Access Port* for more information.

Figure 2-7 on page 2-6 shows the external connections on the AHB-AP.

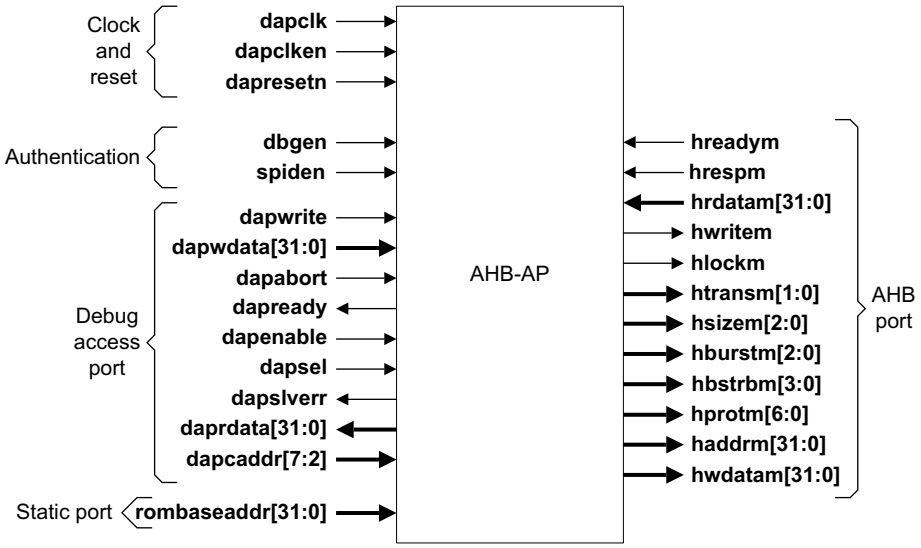


Figure 2-7 Advanced High-performance Bus - Access Port block diagram

2.1.8 Advanced Peripheral Bus Access Port

The *Advanced Peripheral Bus Access Port* (APB-AP) connects a DAP to an APB system bus. The APB-AP has no configurable features.

See Chapter 4 *Debug Access Port* for more information.

Figure 2-8 shows the external connections on the APB-AP.

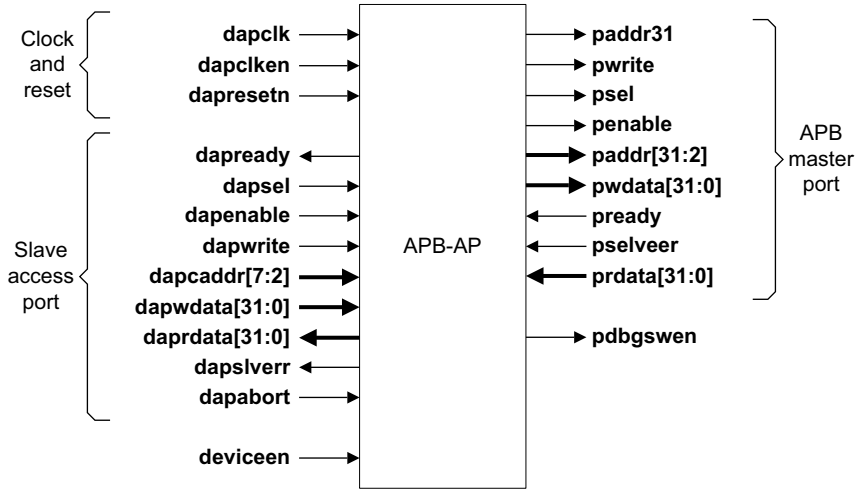


Figure 2-8 Advanced Peripheral Bus - Access Port block diagram

2.1.9 APB Interconnect with ROM table

The *APB InterConnect* (APBIC) with ROM table connects multiple APB masters to multiple slaves. The APBIC implements a ROM table that contains information about the components in a CoreSight system.

The APBIC operates in a single clock domain. Use asynchronous bridges to connect other components that are not synchronous.

The following configurable options must be set for the APBIC:

- SLAVE_INTF_ADDR_WIDTH. See saw in Figure 2-9.
- MASTER_INTF_ADDR_WIDTH. See maw in Figure 2-9.

Figure 2-9 shows the external connections on the APBIC. <x> in the figure denotes an automatically-generated numeric interface number.

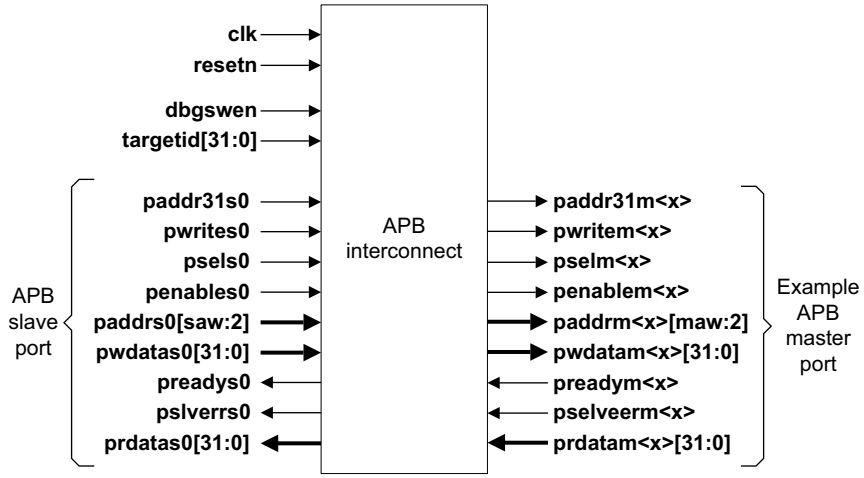


Figure 2-9 APB interconnect with ROM table block diagram

For information about the user-programmable registers for the APBIC, see Chapter 3 *Programmers Model*.

Cascading APBICs

Systems that require more than the maximum configurable number of slaves can use a cascading approach. You can connect two or more APBICs to implement a hierarchy of APB peripherals.

2.1.10 APB asynchronous bridge

The APB asynchronous bridge enables data transfer between two asynchronous clock domains.

The AMBA-compliant LPI is optional on the APB asynchronous bridge.

Figure 2-10 on page 2-8 shows the external connections to the APB asynchronous bridge.

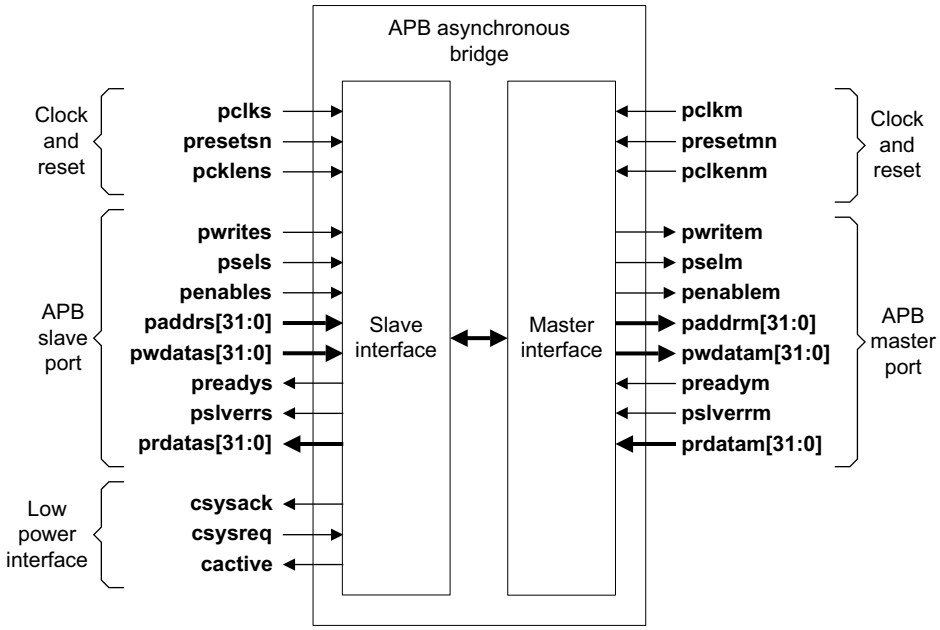


Figure 2-10 APB asynchronous bridge block diagram

2.1.11 APB synchronous bridge

The APB synchronous bridge enables data transfer between two synchronous clock domains.

The AMBA-compliant LPI is optional on the APB synchronous bridge.

Figure 2-11 shows the external connections on the APB synchronous bridge.

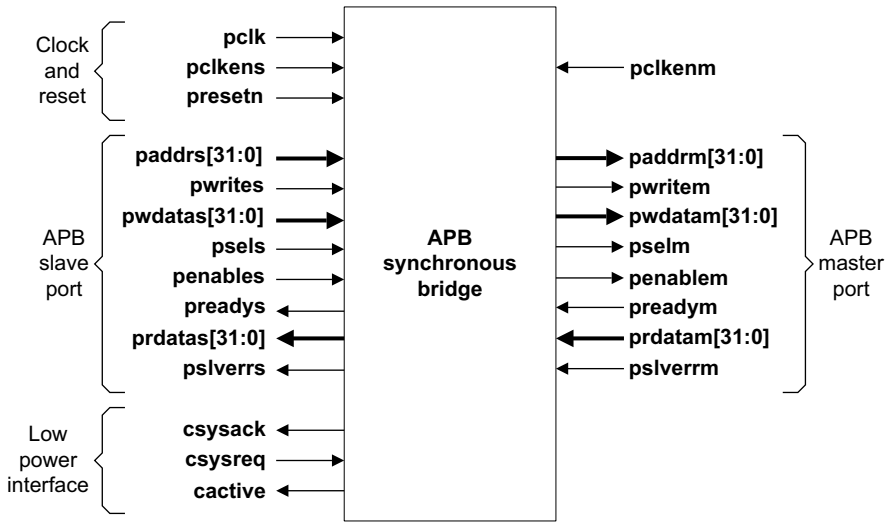


Figure 2-11 APB synchronous bridge block diagram

2.2 Advanced Trace Bus interconnect components

The *Advanced Trace Bus* (ATB) interconnect facilitates the transfer of trace data around the CoreSight debug system. A custom-generated interconnect infrastructure also uses these components to provide aditional functionality as required by your system architecture:

- *ATB replicator*.
- *ATB funnel* on page 2-10.
- *ATB upsizer* on page 2-10.
- *ATB downsizer* on page 2-11.
- *ATB asynchronous bridge* on page 2-11.
- *ATB synchronous bridge* on page 2-12.

See Chapter 5 *ATB Interconnect Components* for more information about these components.

2.2.1 ATB replicator

The ATB replicator propagates data from a single master to two slaves.

Figure 2-12 shows the external connections on the ATB replicator.

You can specify an optional APB configuration interface.

The following parameter is used to implement the ATB replicator:

- `ATB_DATA_WIDTH` of 8, 16, 32, or 64. See `dw` in Figure 2-12 where `dw=ATB_DATA_WIDTH-1`.

Note

In Figure 2-12, *bw* is generated automatically from the parameter `ATB_DATA_WIDTH`.

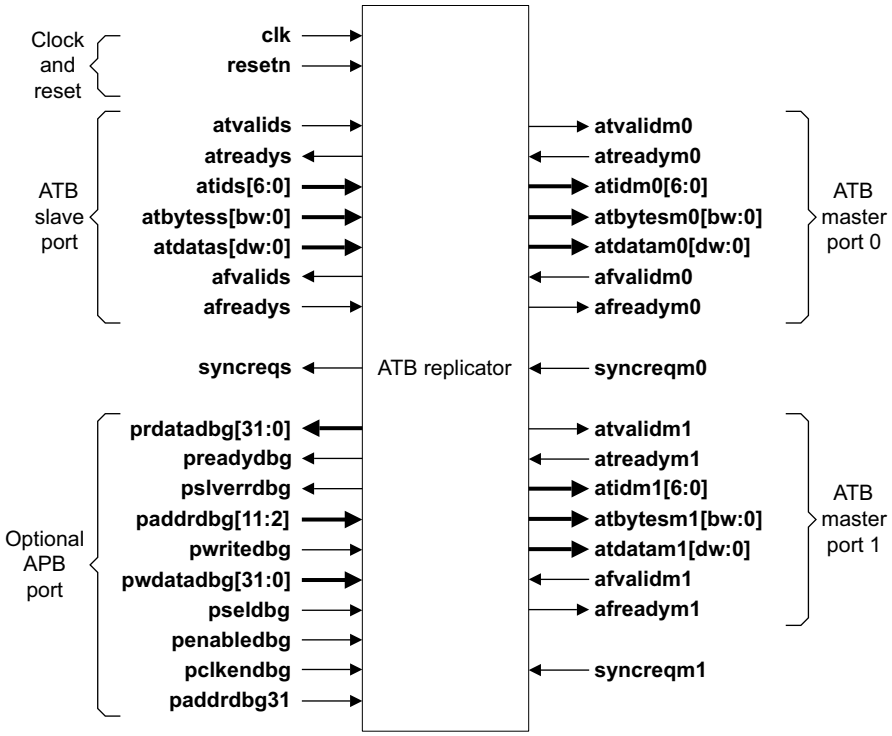


Figure 2-12 ATB replicator block diagram

2.2.2 ATB funnel

The ATB funnel merges the trace from multiple ATB buses and sends the data to a single ATB bus.

Figure 2-13 shows the external connections on the ATB funnel. The <*x*> denotes the auto-generated interface number of the specific ATB interface.

You can specify an optional APB configuration interface.

The following parameter is used to implement the ATB replicator:

- `CXFUNNEL_ATB_DATA_WIDTH` of 8, 16, 32, or 64. See `dw` in Figure 2-13 where `dw=CXFUNNEL_ATB_DATA_WIDTH-1`.

Note

In Figure 2-13, *bw* is generated automatically from the parameter `CXFUNNEL_ATB_DATA_WIDTH`.

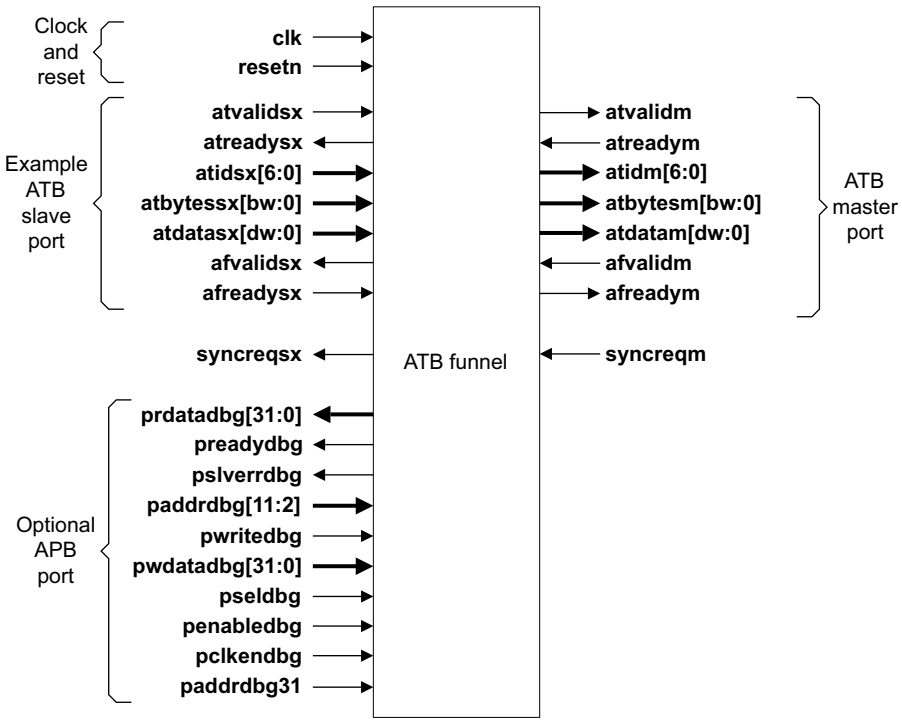


Figure 2-13 ATB funnel block diagram

2.2.3 ATB upsizer

The ATB upsizer converts the trace data on a narrower ATB bus on to a wider bus.

Figure 2-14 on page 2-11 shows the external connections to the ATB upsizer.

The following parameters are used to implement the ATB replicator:

- `ATB_DATA_WIDTH_SLAVE` of 8, 16, 32, or 64. See `sdw` in Figure 2-14 on page 2-11 where `sdw=ATB_DATA_WIDTH_SLAVE-1`.
- `ATB_DATA_WIDTH_MASTER` of 8, 16, 32, or 64. See `mdw` in Figure 2-14 on page 2-11 where `mdw=ATB_DATA_WIDTH_MASTER-1`.

Note

In Figure 2-14, *sbw* and *mbw* are generated automatically from the parameters ATB_DATA_WIDTH_SLAVE and ATB_DATA_WIDTH_MASTER respectively.

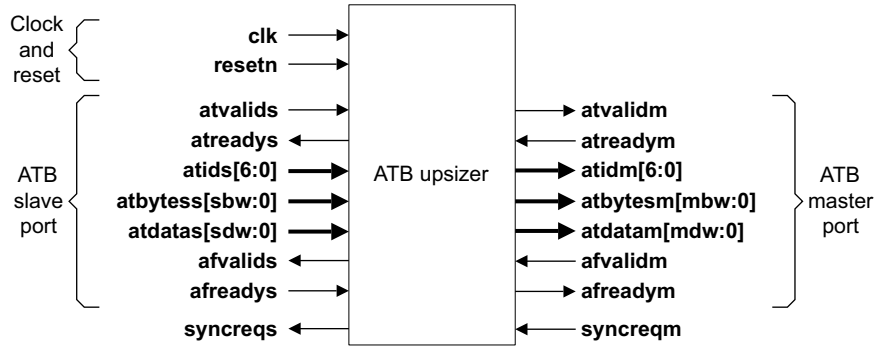


Figure 2-14 ATB upsizer block diagram

2.2.4 ATB downsizer

The ATB downsizer converts the trace data on a wider ATB bus onto a narrower width bus.

Figure 2-15 shows the external connections on the ATB downsizer.

The following parameters are used to implement the ATB replicator:

- ATB_DATA_WIDTH_SLAVE of 8, 16, 32, or 64. See sdw in Figure 2-14 where sdw=ATB_DATA_WIDTH_SLAVE-1.
- ATB_DATA_WIDTH_MASTER of 8, 16, 32, or 64. See mdw in Figure 2-14 where mdw=ATB_DATA_WIDTH_MASTER-1.

Note

In Figure 2-15, *sbw* and *mbw* are generated automatically from the parameters ATB_DATA_WIDTH_SLAVE and ATB_DATA_WIDTH_MASTER respectively.

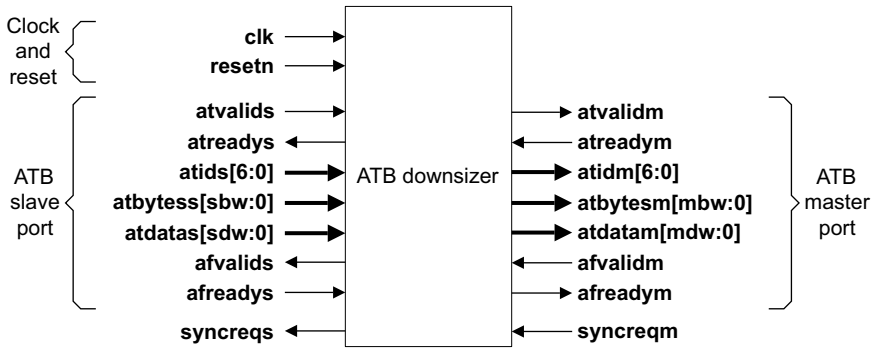


Figure 2-15 ATB downsizer block diagram

2.2.5 ATB asynchronous bridge

The ATB asynchronous bridge enables data transfer between two asynchronous clock domains.

Figure 2-16 on page 2-12 shows the external connections to the ATB asynchronous bridge.

The following parameter is used to implement the ATB replicator:

- `ATB_DATA_WIDTH` of 8, 16, 32, or 64. See `dw` in Figure 2-16 where `dw=ATB_DATA_WIDTH-1`.

Note

In Figure 2-16, *bw* is generated automatically from the parameter `ATB_DATA_WIDTH`.

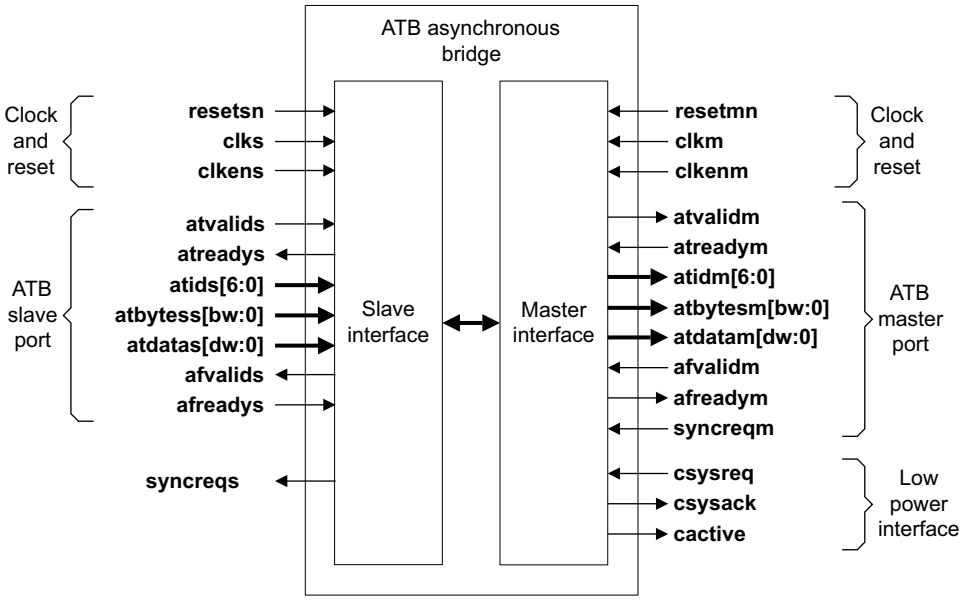


Figure 2-16 ATB asynchronous bridge block diagram

2.2.6 ATB synchronous bridge

The ATB synchronous bridge enables data transfer between two synchronous clock domains.

Figure 2-17 on page 2-13 shows the external connections to the ATB synchronous bridge.

The following parameter is used to implement the ATB replicator:

- `ATB_DATA_WIDTH` of 8, 16, 32, or 64. See `dw` in Figure 2-17 on page 2-13 where `dw=ATB_DATA_WIDTH-1`.

Note

In Figure 2-17 on page 2-13, *bw* is generated automatically from the parameter `ATB_DATA_WIDTH`.

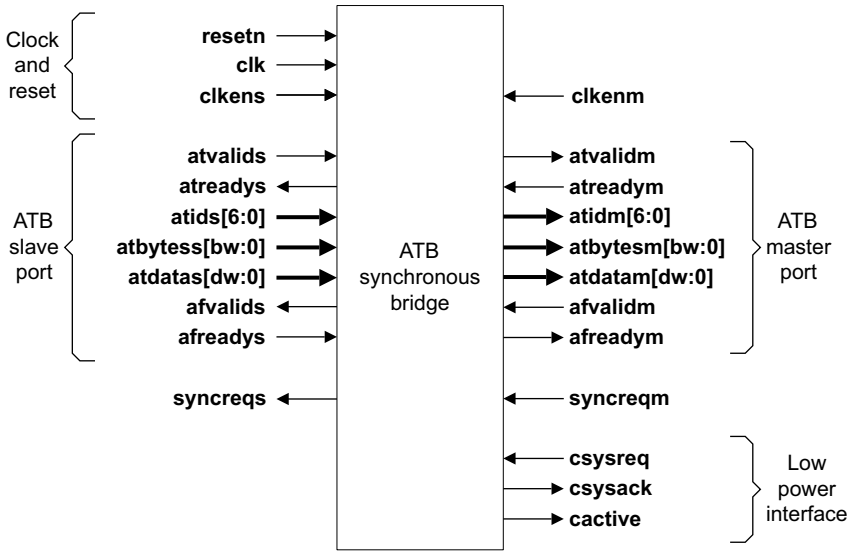


Figure 2-17 ATB synchronous bridge block diagram

2.3 Timestamp components

The timestamp components generate and distribute a consistent timestamp value for multiple processors and other blocks within a SoC. The components available to build this system are:

- *Timestamp generator*.
- *Timestamp encoder*.
- *Narrow timestamp replicator* on page 2-15.
- *Narrow timestamp asynchronous bridge* on page 2-15.
- *Narrow timestamp synchronous bridge* on page 2-16.
- *Timestamp decoder* on page 2-16.
- *Timestamp interpolator* on page 2-17.

See Chapter 6 *Timestamp Components* for more information.

2.3.1 Timestamp generator

The timestamp generator generates a timestamp value that provides a consistent view of time for multiple processors and other blocks in a SoC.

The timestamp generator has no configurable features. For more informaiton on the timestamp generator. See Chapter 6 *Timestamp Components*. For information on the timestamp generator register description, see *Timestamp generator register summary* on page 3-210.

Figure 2-18 shows the external connections on the timestamp generator.

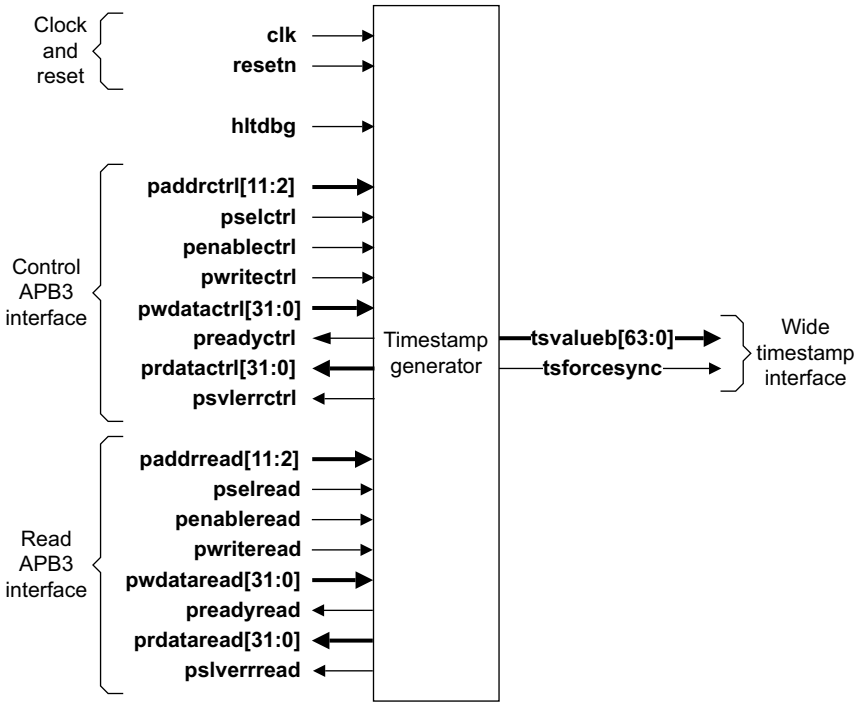


Figure 2-18 Timestamp generator block diagram

2.3.2 Timestamp encoder

The timestamp encoder converts the 64-bit timestamp value from the timestamp generator to a 7-bit encoded value. This is called a narrow timestamp. It also encodes and sends the timestamp value over a 2-bit synchronization channel.

The timestamp encoder has no configurable features.

Figure 2-19 shows the external connections on the timestamp encoder.

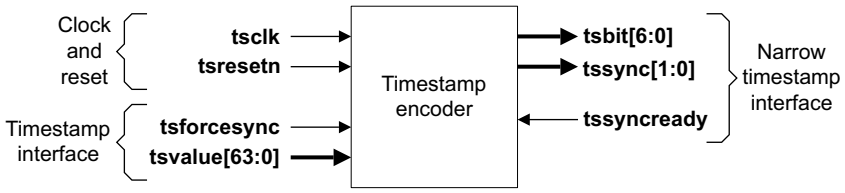


Figure 2-19 Timestamp encoder block diagram

2.3.3 Narrow timestamp replicator

The narrow timestamp replicator distributes the encoded timestamp and synchronization data to multiple slave bridges. You can configure the number of slave bridges to be connected to the narrow timestamp replicator.

Figure 2-20 shows the external connections on the narrow timestamp replicator.

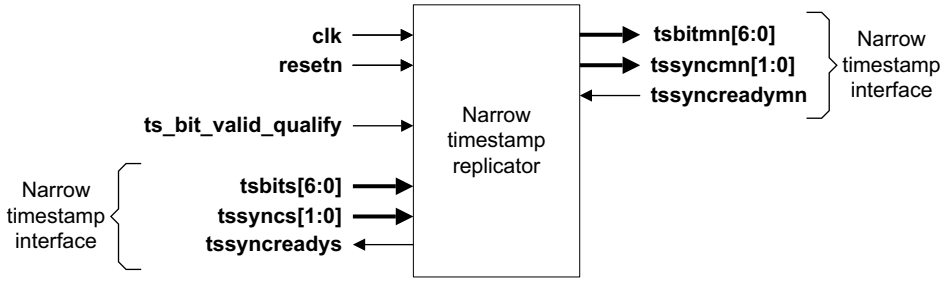


Figure 2-20 Narrow timestamp replicator block diagram

2.3.4 Narrow timestamp asynchronous bridge

The narrow timestamp asynchronous bridge enables the transfer of timestamp information across different clock domains.

Figure 2-21 on page 2-16 shows the external connections on the narrow timestamp asynchronous bridge.

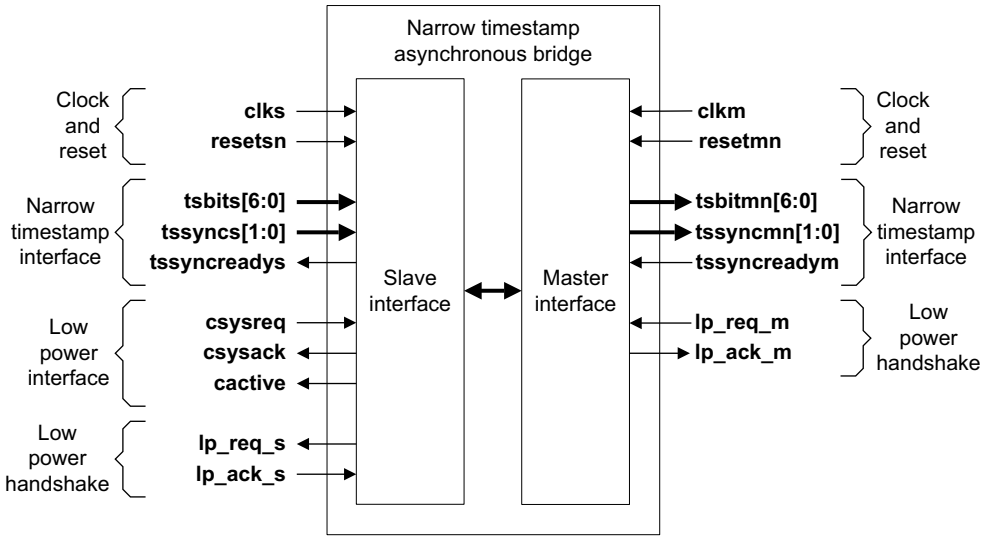


Figure 2-21 Narrow timestamp asynchronous bridge block diagram

2.3.5 Narrow timestamp synchronous bridge

The narrow timestamp synchronous bridge enables the transfer of timestamp information across clock domains that have individual clock enables.

Figure 2-22 shows the external connections on the narrow timestamp synchronous bridge.

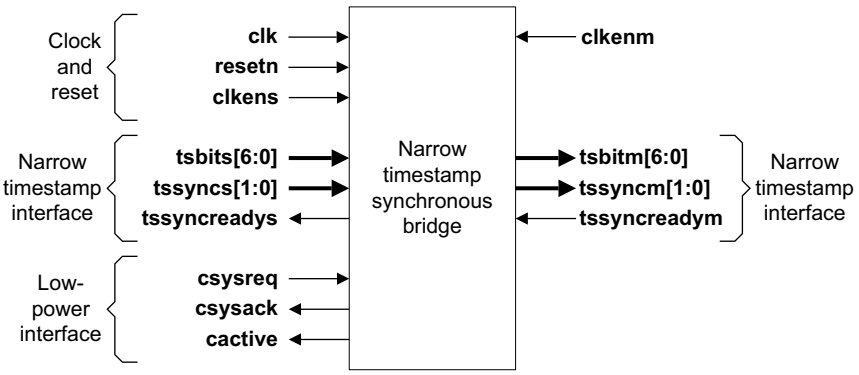


Figure 2-22 Narrow timestamp synchronous bridge block diagram

2.3.6 Timestamp decoder

The timestamp decoder converts the encoded timestamp and synchronization data back to a 64-bit value. This is the format in which the CoreSight trace components require their timestamp.

Figure 2-23 on page 2-17 shows the external connections on the timestamp decoder.

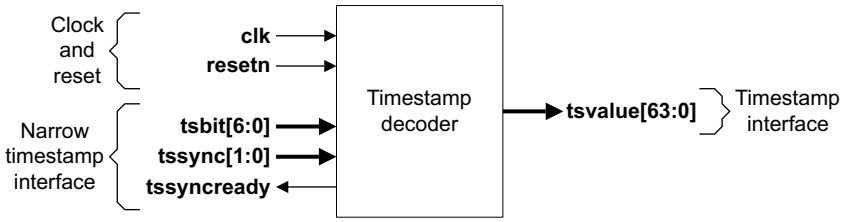


Figure 2-23 Timestamp decoder block diagram

2.3.7 Timestamp interpolator

CoreSight components require timestamp values that allow software to correlate events. A timestamp generator generates timestamp values. This is typically at a clock rate that is much slower than the operating frequency of CoreSight components. The timestamp interpolator uses the timestamp values from the timestamp generator as reference and generates timestamp values at a rate required by CoreSight components.

Figure 2-24 shows the external connections on the timestamp interpolator.

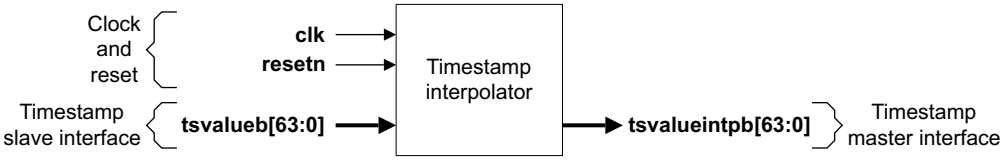


Figure 2-24 Timestamp interpolator block diagram

2.4 Trigger components

CoreSight SoC contains the following trigger components to control the logging of debug information:

- *Cross Trigger Interface*.
- *Cross Trigger Matrix*.

The *Cross Trigger Interface* (CTI) and *Cross Trigger Matrix* (CTM) form the *Embedded Cross Trigger* (ECT) sub-system that enables the ARM *Embedded Trace Macrocell* (ETM) subsystems to interact, that is cross trigger, with each other. The main function of the ECT is to pass debug events from one processor to another. For example, the ECT can communicate debug state information from one processor to the others, so that you can stop the program execution on one or more processors at the same time if required.

2.4.1 Cross Trigger Interface

The CTI combines and maps the trigger requests, and broadcasts them to all other interfaces on the ECT sub-system. When the CTI receives a trigger request it maps this onto a trigger output. This enables the ETM subsystems to cross trigger with each other. Figure 2-25 shows the external connections on the CTI.

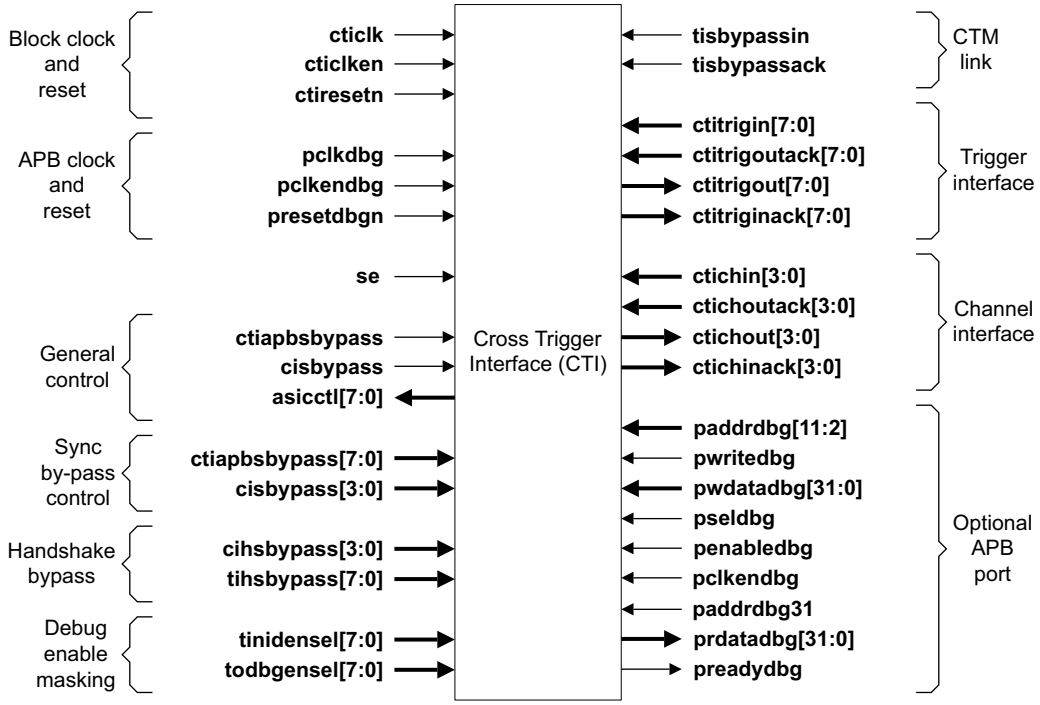


Figure 2-25 Cross Trigger Interface block diagram

2.4.2 Cross Trigger Matrix

The CTM block controls the distribution of trigger requests. It connects to at least two CTIs and to other CTMs where required in a design. Figure 2-26 on page 2-19 shows the external connections on the CTM block.

You must configure the CTM with the following parameter that defines the width of some ports of the block:

- `ECTCHANNELWIDTH`. See `CW` in Figure 2-26 on page 2-19 where `CW=ECTCHANNELWIDTH-1`.

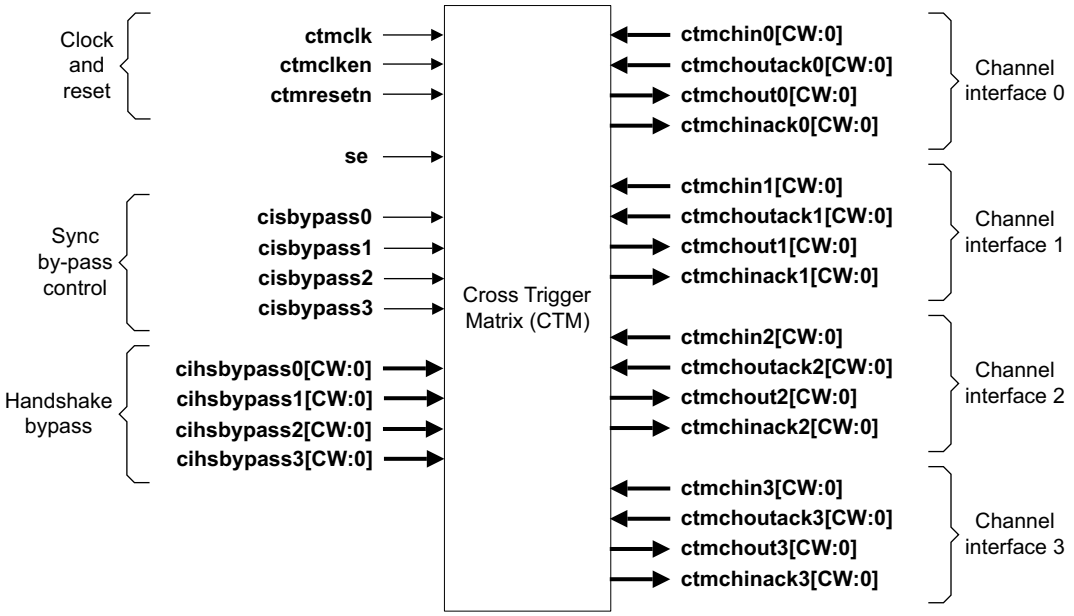


Figure 2-26 Cross Trigger Matrix block diagram

2.5 Trace sink components

CoreSight SoC contains the following components for receiving debug information and looking after its transmission onto the main debug infrastructure. The trace sink components are:

- *Trace Port Interface Unit*.
- *Embedded Trace Buffer*.

2.5.1 Trace Port Interface Unit

The *Trace Port Interface Unit* (TPIU) provides connectivity from an *Advanced Trace Bus* (ATB) to an external trace port. Figure 2-27 shows the external connections on the TPIU.

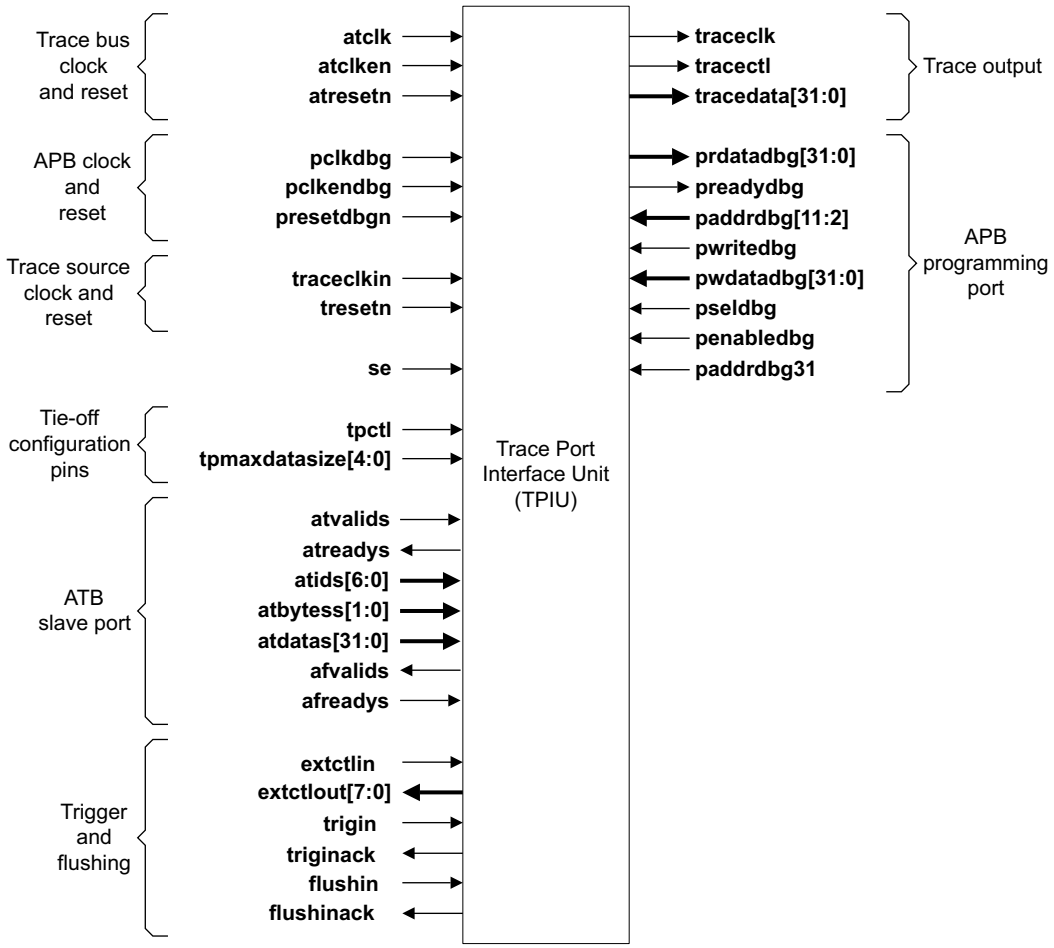


Figure 2-27 Trace Port Interface Unit block diagram

2.5.2 Embedded Trace Buffer

The *Embedded Trace Buffer* (ETB) stores trace data while waiting for transmission onto the CoreSight system. Figure 2-28 on page 2-21 shows the external connections on the ETB.

You must configure the ETB with the following parameter that defines the width of the MBIST address for trace RAM testing:

- `CSETB_RAM_ADRW`. See `AW` in Figure 2-28 on page 2-21 where `AW=CSETB_RAM_ADRW-1`.

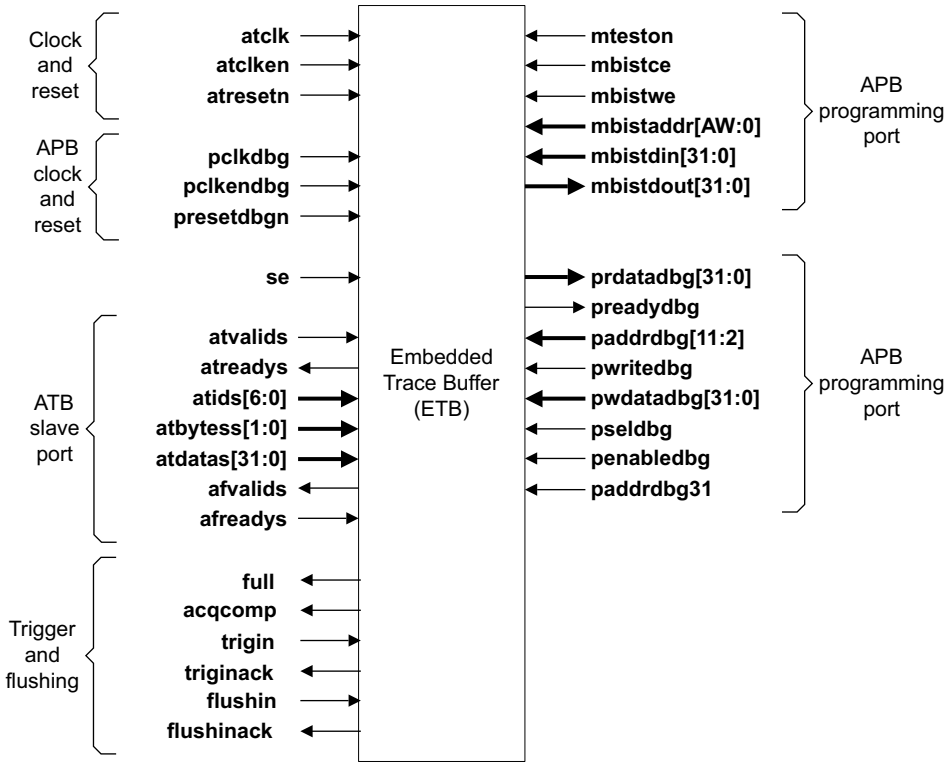


Figure 2-28 Embedded Trace Buffer block diagram

2.6 Authentication and event bridges

The additional bridges are:

- *Authentication asynchronous bridge*.
- *Authentication synchronous bridge*.
- *Event asynchronous bridge* on page 2-23.

The authentication bridges provide authenticated debug control links in security-enabled CoreSight systems. These components are not required if this security is not required.

2.6.1 Authentication asynchronous bridge

The authentication asynchronous bridge enables transfers of authentication signals between two asynchronous clock domains.

You must configure the authentication asynchronous bridge during implementation, with the authentication signals that are present.

Figure 2-29 shows the external connections on the authentication asynchronous bridge.

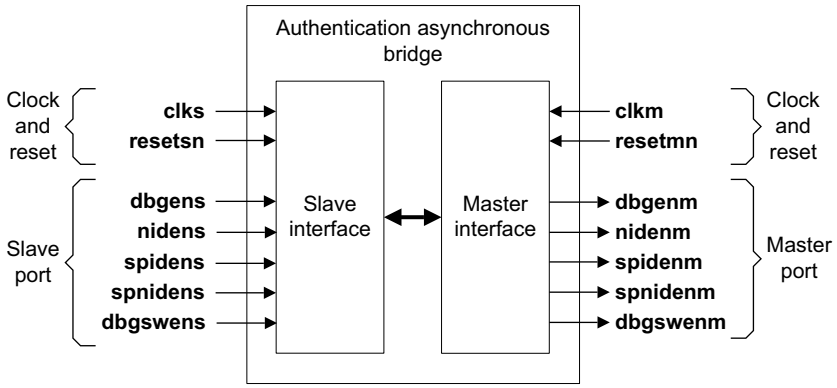


Figure 2-29 Authentication asynchronous bridge block diagram

2.6.2 Authentication synchronous bridge

The authentication synchronous bridge enables the transfers of authentication signals between two synchronous clock domains. It can also be used as a register slice to break long timing paths.

You must configure the authentication synchronous bridge during implementation, with the authentication signals that are present.

Figure 2-30 shows the external connections on the authentication synchronous bridge.

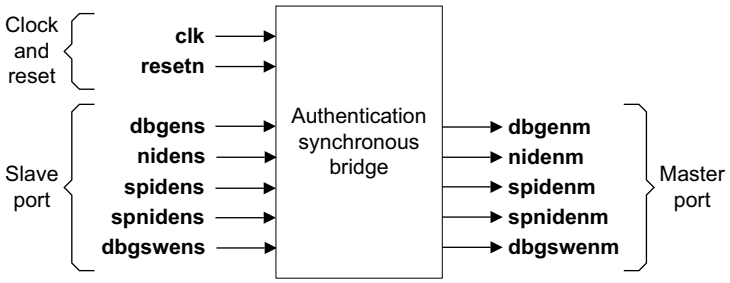


Figure 2-30 Authentication synchronous bridge block diagram

2.6.3 Event asynchronous bridge

The event asynchronous bridge is a fixed component that synchronizes events on a single channel from the slave domain to the master domain. The event acknowledge from the master domain is synchronized and presented to the slave domain.

If the event is a pulse, the bridge internally stretches the event until it receives an acknowledge from the master domain. In this mode of operation, additional events from the slave domain are ignored until the acknowledge is received at the slave domain.

Figure 2-31 shows the external connections on the event asynchronous bridge.

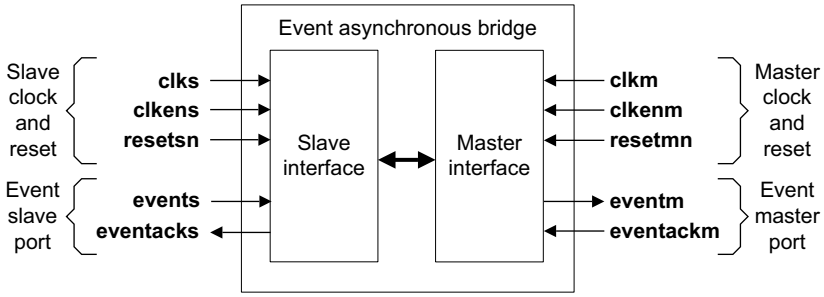


Figure 2-31 Event asynchronous bridge block diagram

This component supports the following combinations of asynchronous event interface and synchronous event interface. In a synchronous event interface, the event signal, when changed, can be changed again without waiting for the assertion of the **eventack** signal. In an asynchronous event interface, the event signal, when changed, can be changed again only after the acknowledgement on the **eventack** signal.

Asynchronous slave interface and asynchronous master interface

No specific connectivity considerations.

Asynchronous slave interface and synchronous master interface

The input port, **eventackm**, must be tied HIGH.

Synchronous slave interface and asynchronous master interface

The output port, **eventacks**, can be left unconnected.

Synchronous slave interface and synchronous master interface

The input port, **eventackm**, must be tied HIGH, and the output port, **eventacks**, can be left unconnected.

2.7 Granular Power Requestor

The *Granular Power Requester* (GPR) enables a debugger to control powerup and powerdown of specific components within a CoreSight system. Without the GPR, the CoreSight DAP components only support system level powerup and powerdown of the entire CoreSight system. The finer granularity provided by the GPR enables implementation of power strategies during both debug and ATPG testing.

Figure 2-32 shows the external connections on the GPR.

You must configure the GPR during implementation with the following parameters:

- `NUM_CPWRUPM`. See Figure 2-32.

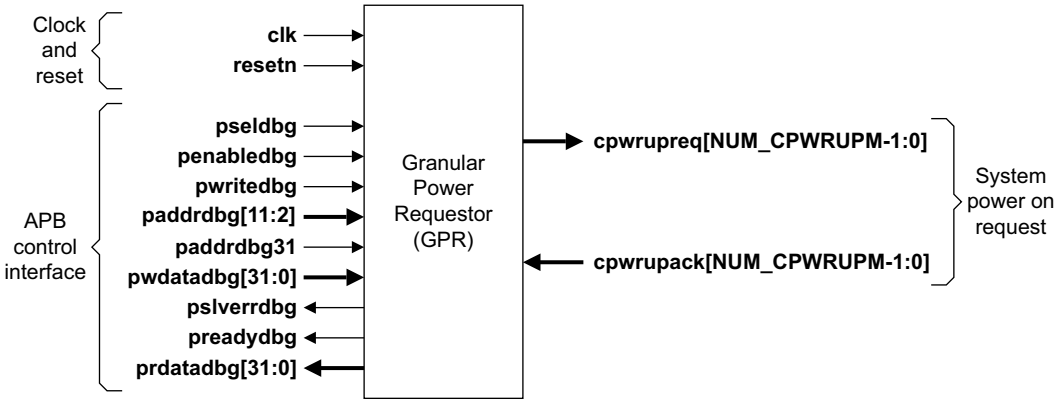


Figure 2-32 Granular Power Requestor block diagram

The GPR has the following properties:

- The interfaces comply with the APB3 AMBA protocol and *CoreSight Architecture Specification*.
- The GPR operates in a single debug power domain.
- Up to 32 power domains are supported.

Chapter 3
Programmers Model

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

3.1 About the programmers model

This section describes register information for the CoreSight SoC components.

The following applies to all the CoreSight SoC register descriptions:

- All CoreSight component registers are 32 bits wide.
- 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 locations can result in UNPREDICTABLE behavior.
- Unless otherwise stated in the accompanying text:
  - Do not modify UNDEFINED register bits.
  - Ignore UNDEFINED register bits on a read operation.
  - A system or power-on reset resets all register bits to logic 0.
- Access types are described as follows:

RW	Read and write.
RO	Read-only.
WO	Write-only.
SBZ	Should-Be-Zero.
SBZP	Should-Be-Zero-or-Preserved.
RAZ	Read-As-Zero.
RAZ/WI	Read-As-Zero, Writes Ignored.

3.2 *Granular Power Requestor* (GPR) register summary

Table 3-1 shows the GPR registers in offset order from the base memory address.

Table 3-1 GPR register summary

Offset	Name	Type	Reset	Description
0x000	CPWRUPREQ	RW	0x00000000	*Debug Power Request Register* on page 3-4
0x004	CPWRUPACK	RO	0x00000000	*Debug Power Acknowledge Register* on page 3-8
0xF00	ITCTRL	RO	0x00000000	*Integration Mode Control Register* on page 3-12
0xFA0	CLAIMSET	RW	0x0000000F	*Claim Tag Set Register* on page 3-13
0xFA4	CLAIMCLR	RW	0x00000000	*Claim Tag Clear Register* on page 3-13
0xFB0	LAR	WO	0x00000000	*Lock Access Register* on page 3-14
0xFB4	LSR	RO	0x00000003	*Lock Status Register* on page 3-15
0xFB8	AUTHSTATUS	RO	0x00000000	*Authentication Status Register* on page 3-15
0xFBC	DEVARCH	RO	0x00000000	*Device Architecture Register* on page 3-16
0xFC8	DEVID	RO	0x00000001	*Device Configuration Register* on page 3-17
0xFCC	DEVTYPE	RO	0x00000034	*Device Type Identifier Register* on page 3-17
0xFD0	PIDR4	RO	0x00000004	*Peripheral ID4 Register* on page 3-18
0xFD4	PIDR5	RO	0x00000000	*Peripheral ID5-7 registers* on page 3-19
0xFD8	PIDR6	RO	0x00000000	
0xFDC	PIDR7	RO	0x00000000	
0xFE0	PIDR0	RO	0x000000A4	*Peripheral ID0 Register* on page 3-19
0xFE4	PIDR1	RO	0x000000B9	*Peripheral ID1 Register* on page 3-20
0xFE8	PIDR2	RO	0x0000000B	*Peripheral ID2 Register* on page 3-21
0xFEC	PIDR3	RO	0x00000000	*Peripheral ID3 Register* on page 3-21
0xFF0	CIDR0	RO	0x0000000D	*Component ID0 Register* on page 3-22
0xFF4	CIDR1	RO	0x00000090	*Component ID1 Register* on page 3-23
0xFF8	CIDR2	RO	0x00000005	*Component ID2 Register* on page 3-23
0xFFC	CIDR3	RO	0x000000B1	*Component ID3 Register* on page 3-24

3.3 GPR register descriptions

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

3.3.1 Debug Power Request Register

The CPWRUPREQ Register characteristics are:

Purpose Controls the values of the **cpwrupreq** outputs from CXGPR. Each bit in this register controls the corresponding output on the **cpwrupreq** port. CXGPR contains hardware logic to ensure that the 4-phase handshake is not violated on the CPWRUP interfaces.

If CXGPR asserts a powerup request that is not acknowledged, that is, **cpwrupreq[n]** = 1 and **cpwrupack[n]** = 0, writing a 0 to the CPWRUPREQ register bit[n] does not affect the **cpwrupreq[n]** output.

Similarly, if CXGPR sends a powerdown request that is not acknowledged, that is, **cpwrupreq[n]** is 0 and **cpwrupack[n]** = 1, writing a 1 to the CPWRUPREQ register bit[n] does not affect the **cpwrupreq[n]** output.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-1 on page 3-3.

Figure 3-1 on page 3-5 shows the bit assignments.

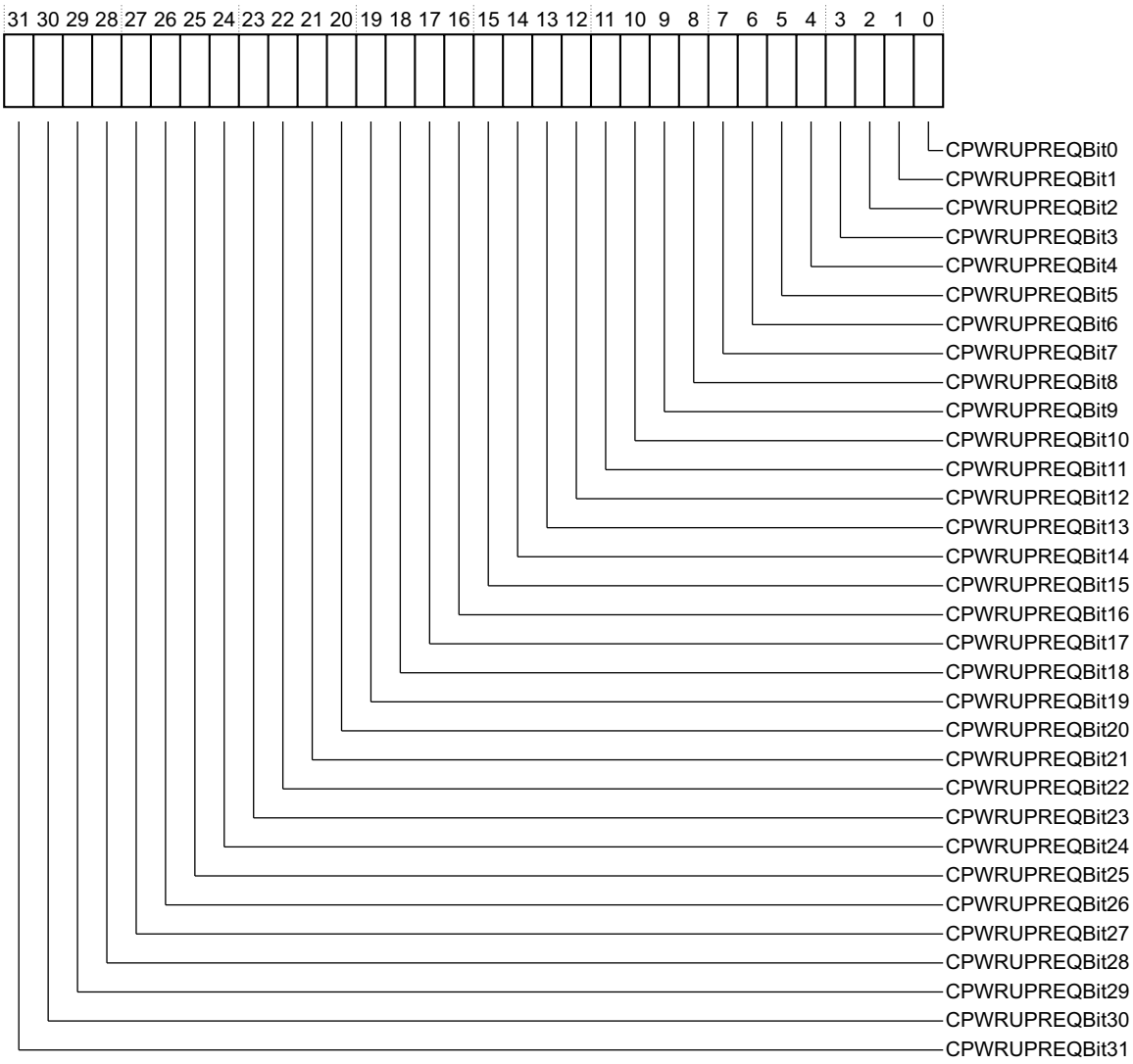


Figure 3-1 CPWRUPREQ Register bit assignments

Table 3-2 shows the bit assignments.

Table 3-2 CPWRUPREQ Register bit assignments

Bits	Name	Function
[31]	CPWRUPREQBit31	Bit[31] of the **cpwrupreq** output port. `0b0` Drive 0 on **cpwrupreq[31]** output port. `0b1` Drive 1 on **cpwrupreq[31]** output port.
[30]	CPWRUPREQBit30	Bit[30] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[30]** output port. `0b1` Drive 1 on **cpwrupreq[30]** output port.
[29]	CPWRUPREQBit29	Bit[29] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[29]** output port. `0b1` Drive 1 on **cpwrupreq[29]** output port.

Table 3-2 CPWRUPREQ Register bit assignments (continued)

Bits	Name	Function
[28]	CPWRUPREQBit28	Bit[28] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[28]** output port. `0b1` Drive 1 on **cpwrupreq[28]** output port.
[27]	CPWRUPREQBit27	Bit[27] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[27]** output port. `0b1` Drive 1 on **cpwrupreq[27]** output port.
[26]	CPWRUPREQBit26	Bit[26] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[26]** output port. `0b1` Drive 1 on **cpwrupreq[26]** output port.
[25]	CPWRUPREQBit25	Bit[25] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[25]** output port. `0b1` Drive 1 on **cpwrupreq[25]** output port.
[24]	CPWRUPREQBit24	Bit[24] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[24]** output port. `0b1` Drive 1 on **cpwrupreq[24]** output port.
[23]	CPWRUPREQBit23	Bit[23] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[23]** output port. `0b1` Drive 1 on **cpwrupreq[23]** output port.
[22]	CPWRUPREQBit22	Bit[22] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[22]** output port. `0b1` Drive 1 on **cpwrupreq[22]** output port.
[21]	CPWRUPREQBit21	Bit[21] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[21]** output port. `0b1` Drive 1 on **cpwrupreq[21]** output port.
[20]	CPWRUPREQBit20	Bit[20] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[20]** output port. `0b1` Drive 1 on **cpwrupreq[20]** output port.
[19]	CPWRUPREQBit19	Bit[19] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[19]** output port. `0b1` Drive 1 on **cpwrupreq[19]** output port.
[18]	CPWRUPREQBit18	Bit[18] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[18]** output port. `0b1` Drive 1 on **cpwrupreq[18]** output port.
[17]	CPWRUPREQBit17	Bit[17] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[17]** output port. `0b1` Drive 1 on **cpwrupreq[17]** output port.
[16]	CPWRUPREQBit16	Bit[16] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[16]** output port. `0b1` Drive 1 on **cpwrupreq[16]** output port.

Table 3-2 CPWRUPREQ Register bit assignments (continued)

Bits	Name	Function
[15]	CPWRUPREQBit15	Bit[15] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[15]** output port. `0b1` Drive 1 on **cpwrupreq[15]** output port.
[14]	CPWRUPREQBit14	Bit[14] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[14]** output port. `0b1` Drive 1 on **cpwrupreq[14]** output port.
[13]	CPWRUPREQBit13	Bit[13] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[13**] output port. `0b1` Drive 1 on **cpwrupreq[13]** output port.
[12]	CPWRUPREQBit12	Bit[12] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[12]** output port. `0b1` Drive 1 on **cpwrupreq[12]** output port.
[11]	CPWRUPREQBit11	Bit[11] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[11]** output port. `0b1` Drive 1 on **cpwrupreq[11]** output port.
[10]	CPWRUPREQBit10	Bit[10] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[10]** output port. `0b1` Drive 1 on **cpwrupreq[10]** output port.
[9]	CPWRUPREQBit9	Bit[9] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[9]** output port. `0b1` Drive 1 on **cpwrupreq[9]** output port.
[8]	CPWRUPREQBit8	Bit[8] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[8]** output port. `0b1` Drive 1 on **cpwrupreq[8]** output port.
[7]	CPWRUPREQBit7	Bit[7] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[7]** output port. `0b1` Drive 1 on **cpwrupreq[7]** output port.
[6]	CPWRUPREQBit6	Bit[6] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[6]** output port. `0b1` Drive 1 on **cpwrupreq[6]** output port.
[5]	CPWRUPREQBit5	Bit[5] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[5]** output port. `0b1` Drive 1 on **cpwrupreq[5]** output port.
[4]	CPWRUPREQBit4	Bit[4] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[4]** output port. `0b1` Drive 1 on **cpwrupreq[4]** output port.
[3]	CPWRUPREQBit3	Bit[3] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[3]** output port. `0b1` Drive 1 on **cpwrupreq[3]** output port.

Table 3-2 CPWRUPREQ Register bit assignments (continued)

Bits	Name	Function
[2]	CPWRUPREQBit2	Bit[2] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[2]** output port. `0b1` Drive 1 on **cpwrupreq[2]** output port.
[1]	CPWRUPREQBit1	Bit[1] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[1]** output port. `0b1` Drive 1 on **cpwrupreq[1]** output port.
[0]	CPWRUPREQBit0	Bit[0] of the **cpwrupreq** output port: `0b0` Drive 0 on **cpwrupreq[0]** output port. `0b1` Drive 1 on **cpwrupreq[0]** output port.

3.3.2 Debug Power Acknowledge Register

The CPWRUPACK Register characteristics are:

Purpose Returns the value of the **cpwrupack** input port. Each bit in this register reflects the status of a powerup request.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-1 on page 3-3.

Figure 3-2 on page 3-9 shows the bit assignments.

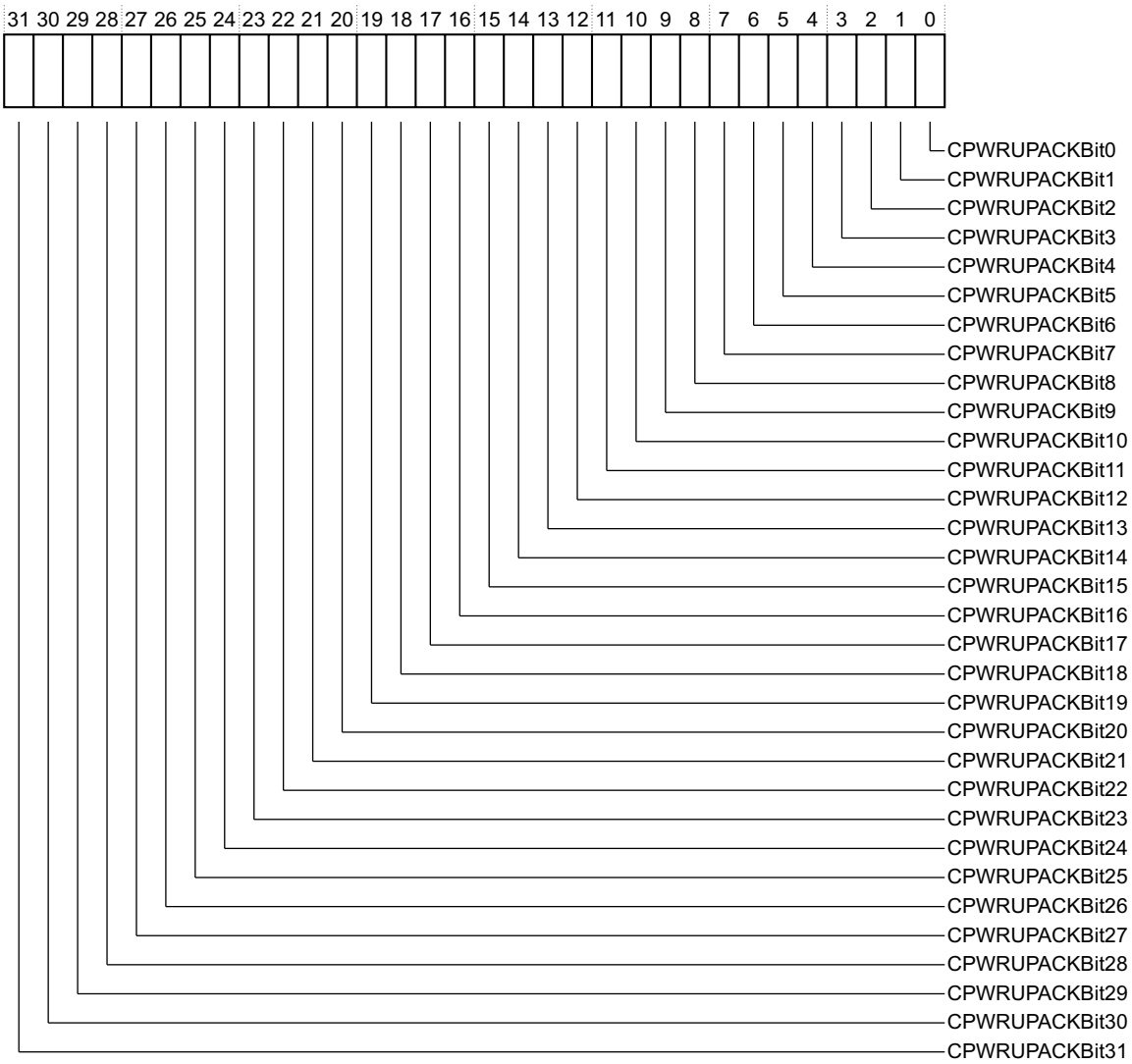


Figure 3-2 CPWRUPACK Register bit assignments

Table 3-3 shows the bit assignments.

Table 3-3 CPWRUPACK Register bit assignments

Bits	Name	Function
[31]	CPWRUPACKBit31	Bit[31] of the **cpwrupack** input port: `0b0` **cpwrupack[31]** input port is LOW. `0b1` **cpwrupack[31]** input port is HIGH.
[30]	CPWRUPACKBit30	Bit[30] of the **cpwrupack** input port: `0b0` **cpwrupack[30]** input port is LOW. `0b1` **cpwrupack[30]** input port is HIGH.
[29]	CPWRUPACKBit29	Bit[29] of the **cpwrupack** input port: `0b0` **cpwrupack[29]** input port is LOW. `0b1` **cpwrupack[29]** input port is HIGH.

Table 3-3 CPWRUPACK Register bit assignments (continued)

Bits	Name	Function
[28]	CPWRUPACKBit28	Bit[28] of the **cpwrupack** input port: `0b0` **cpwrupack[28]** input port is LOW. `0b1` **cpwrupack[28]** input port is HIGH.
[27]	CPWRUPACKBit27	Bit[27] of the **cpwrupack** input port: `0b0` **cpwrupack[27]** input port is LOW. `0b1` **cpwrupack[27]** input port is HIGH.
[26]	CPWRUPACKBit26	Bit[26] of the **cpwrupack** input port: `0b0` **cpwrupack[26]** input port is LOW. `0b1` **cpwrupack[26]** input port is HIGH.
[25]	CPWRUPACKBit25	Bit[25] of the **cpwrupack** input port: `0b0` **cpwrupack[25]** input port is LOW. `0b1` **cpwrupack[25]** input port is HIGH.
[24]	CPWRUPACKBit24	Bit[24] of the **cpwrupack** input port: `0b0` **cpwrupack[24]** input port is LOW. `0b1` **cpwrupack[24]** input port is HIGH.
[23]	CPWRUPACKBit23	Bit[23] of the **cpwrupack** input port: `0b0` **cpwrupack[23]** input port is LOW. `0b1` **cpwrupack[23]** input port is HIGH.
[22]	CPWRUPACKBit22	Bit[22] of the **cpwrupack** input port: `0b0` **cpwrupack[22]** input port is LOW. `0b1` **cpwrupack[22]** input port is HIGH.
[21]	CPWRUPACKBit21	Bit[21] of the **cpwrupack** input port: `0b0` **cpwrupack[21]** input port is LOW. `0b1` **cpwrupack[21]** input port is HIGH.
[20]	CPWRUPACKBit20	Bit[20] of the **cpwrupack** input port: `0b0` **cpwrupack[20]** input port is LOW. `0b1` **cpwrupack[20]** input port is HIGH.
[19]	CPWRUPACKBit19	Bit[19] of the **cpwrupack** input port: `0b0` **cpwrupack[19]** input port is LOW. `0b1` **cpwrupack[19]** input port is HIGH.
[18]	CPWRUPACKBit18	Bit[18] of the **cpwrupack** input port: `0b0` **cpwrupack[18]** input port is LOW. `0b1` **cpwrupack[18]** input port is HIGH.
[17]	CPWRUPACKBit17	Bit[17] of the **cpwrupack** input port: `0b0` **cpwrupack[17]** input port is LOW. `0b1` **cpwrupack[17]** input port is HIGH.
[16]	CPWRUPACKBit16	Bit[16] of the **cpwrupack** input port: `0b0` **cpwrupack[16]** input port is LOW. `0b1` **cpwrupack[16]** input port is HIGH.

Table 3-3 CPWRUPACK Register bit assignments (continued)

Bits	Name	Function
[15]	CPWRUPACKBit15	Bit[15] of the **cpwrupack** input port: 0b0 **cpwrupack[15]** input port is LOW. 0b1 **cpwrupack[15]** input port is HIGH.
[14]	CPWRUPACKBit14	Bit[14] of the **cpwrupack** input port: 0b0 **cpwrupack[14]** input port is LOW. 0b1 **cpwrupack[14]** input port is HIGH.
[13]	CPWRUPACKBit13	Bit[13] of the **cpwrupack** input port: 0b0 **cpwrupack[13]** input port is LOW. 0b1 **cpwrupack[13]** input port is HIGH.
[12]	CPWRUPACKBit12	Bit[12] of the **cpwrupack** input port: 0b0 **cpwrupack[12]** input port is LOW. 0b1 **cpwrupack[12]** input port is HIGH.
[11]	CPWRUPACKBit11	Bit[11] of the **cpwrupack** input port: 0b0 **cpwrupack[11]** input port is LOW. 0b1 **cpwrupack[11]** input port is HIGH.
[10]	CPWRUPACKBit10	Bit[10] of the **cpwrupack** input port: 0b0 **cpwrupack[10]** input port is LOW. 0b1 **cpwrupack[10]** input port is HIGH.
[9]	CPWRUPACKBit9	Bit[9] of the **cpwrupack** input port: 0b0 **cpwrupack[9]** input port is LOW. 0b1 **cpwrupack[9]** input port is HIGH.
[8]	CPWRUPACKBit8	Bit[8] of the **cpwrupack** input port: 0b0 **cpwrupack[8]** input port is LOW. 0b1 **cpwrupack[8]** input port is HIGH.
[7]	CPWRUPACKBit7	Bit[7] of the **cpwrupack** input port: 0b0 **cpwrupack[7]** input port is LOW. 0b1 **cpwrupack[7]** input port is HIGH.
[6]	CPWRUPACKBit6	Bit[6] of the **cpwrupack** input port: 0b0 **cpwrupack[6]** input port is LOW. 0b1 **cpwrupack[6]** input port is HIGH.
[5]	CPWRUPACKBit5	Bit[5] of the **cpwrupack** input port: 0b0 **cpwrupack[5]** input port is LOW. 0b1 **cpwrupack[5]** input port is HIGH.
[4]	CPWRUPACKBit4	Bit[4] of the **cpwrupack** input port: 0b0 **cpwrupack[4]** input port is LOW. 0b1 **cpwrupack[4]** input port is HIGH.
[3]	CPWRUPACKBit3	Bit[3] of the **cpwrupack** input port: 0b0 **cpwrupack[3]** input port is LOW. 0b1 **cpwrupack[3]** input port is HIGH.

Table 3-3 CPWRUPACK Register bit assignments (continued)

Bits	Name	Function
[2]	CPWRUPACKBit2	Bit[2] of the **cpwrupack** input port: `0b0` **cpwrupack[2]** input port is LOW. `0b1` **cpwrupack[2]** input port is HIGH.
[1]	CPWRUPACKBit1	Bit[1] of the **cpwrupack** input port `0b0` **cpwrupack[1]** input port is LOW. `0b1` **cpwrupack[1]** input port is HIGH.
[0]	CPWRUPACKBit0	Bit[0] of the **cpwrupack** input port: `0b0` **cpwrupack[0]** input port is LOW. `0b1` **cpwrupack[0]** input port is HIGH.

3.3.3 Integration Mode Control Register

The ITCTRL Register characteristics are:

Purpose Enables topology detection. For more information, see the *CoreSight Architecture Specification*. 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 detection.

Note

When you read this register, the CXGPR component returns a zero because it has no integration functionality.

After performing integration or topology detection, you must reset the system to ensure correct behavior of CoreSight and other connected system components that the integration or topology detection can affect.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-1 on page 3-3.

Figure 3-3 shows the bit assignments.

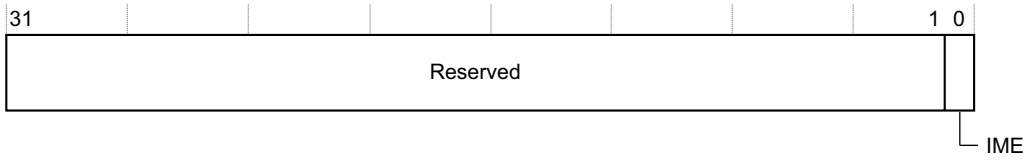


Figure 3-3 ITCTRL Register bit assignments

Table 3-4 shows the bit assignments.

Table 3-4 ITCTRL Register bit assignments

Bits	Name	Function
[31:1]	Reserved	-
[0]	IME	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. **0** Disable integration mode.

3.3.4 Claim Tag Set Register

The CLAIMSET Register characteristics are:

Purpose Software can use the claim tag to coordinate application and debugger access to trace unit functionality. The claim tags have no effect on the operation of the component. The CLAIMSET Register sets bits in the claim tag, and determines the number of claim bits implemented.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-1 on page 3-3.

Figure 3-4 shows the bit assignments.

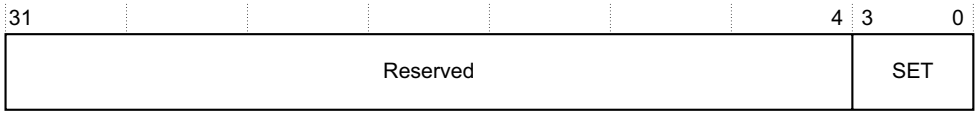


Figure 3-4 CLAIMSET Register bit assignments

Table 3-5 shows the bit assignments.

Table 3-5 CLAIMSET Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	SET_W	When you write 0 to bit[n] of this register, there is no effect on the claim tag bit[n]. When you write 1 to bit[n] of this register, bit[n] in the claim tag is set to 1.
[3:0]	SET_R	When you read 0 on a particular bit of this register, it indicates that this claim tag bit is not implemented. When you read 1 on a particular bit of this register, it indicates that this claim tag bit is implemented. `0b1111` Indicates that four bits of claim tag are implemented.

3.3.5 Claim Tag Clear Register

The CLAIMCLR Register characteristics are:

Purpose Software can use the claim tag to coordinate application and debugger access to trace unit functionality. The claim tags have no effect on the operation of the component. The CLAIMCLR Register clears the bits in the claim tag, and determines the current value of the claim tag.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-1 on page 3-3.

Figure 3-5 shows the bit assignments.

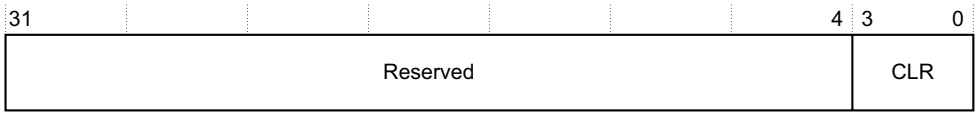


Figure 3-5 CLAIMCLR Register bit assignments

Table 3-6 shows the bit assignments.

Table 3-6 CLAIMCLR Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	CLR_W	When you write 0 to bit[n] of this register, there is no effect on the claim tag bit[n]. When you write 1 to bit[n] of this register, bit[n] in the claim tag is cleared to 0.
[3:0]	CLR_R	The value reflects the present value of the claim tag.

3.3.6 Lock Access Register

The LAR characteristics are:

Purpose Controls write access from self-hosted, on-chip accesses. The LAR does not affect the accesses using the external debugger interface.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-1 on page 3-3.

Figure 3-6 shows the bit assignments.

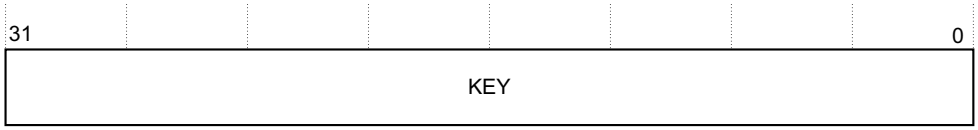


Figure 3-6 LAR bit assignments

Table 3-7 shows the bit assignments.

Table 3-7 LAR bit assignments

Bits	Name	Function
[31:0]	KEY	When you write `0xC5ACCE55`, subsequent write operations to this device are enabled. Any other value disables subsequent write operations.

3.3.7 Lock Status Register

The LSR characteristics are:

Purpose Indicates the status of the lock control mechanism. This lock prevents accidental writes by code. When locked, write accesses are denied for all registers except for the LAR. The lock registers do not affect accesses from the external debug interface. This register reads as 0 when accessed from the external debug interface.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-1 on page 3-3.

Figure 3-7 shows the bit assignments.

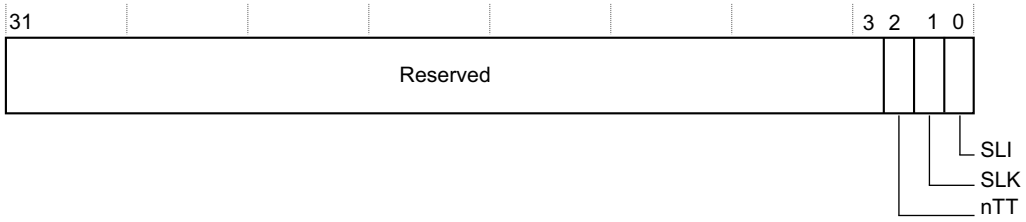


Figure 3-7 LSR bit assignments

Table 3-8 shows the bit assignments.

Table 3-8 LSR bit assignments

Bits	Name	Function
[31:3]	Reserved	-
[2]	nTT	Indicates that the LAR is implemented as 32-bit. `0b0` This component implements a 32-bit LAR.
[1]	SLK	Returns the present lock status of the device. `0b0` Indicates that write operations are permitted in this device. `0b1` Indicates that write operations are not permitted in this device. Read operations are permitted.
[0]	SLI	Indicates that a lock control mechanism is present in this device. `0b0` Indicates that a lock control mechanism is not present in this device. Write operations to the LAR are ignored. `0b1` Indicates that a lock control mechanism is present in this device.

3.3.8 Authentication Status Register

The AUTHSTATUS Register characteristics are:

Purpose Reports the required security level and present status.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-1 on page 3-3.

Figure 3-8 on page 3-16 shows the bit assignments.

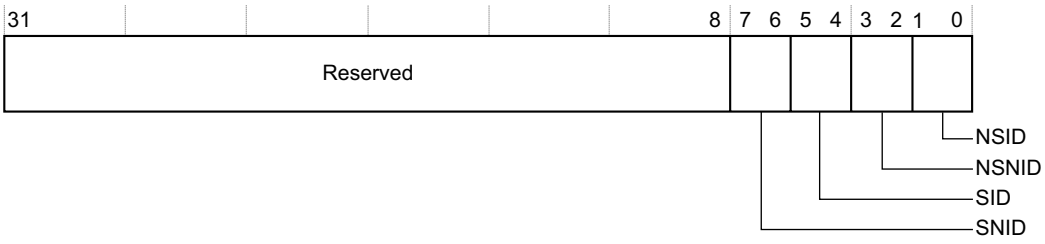


Figure 3-8 AUTHSTATUS Register bit assignments

Table 3-9 shows the bit assignments.

Table 3-9 AUTHSTATUS Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:6]	SNID	Indicates the security level for secure non-invasive debug: `0b00` Functionality is not implemented or is controlled elsewhere.
[5:4]	SID	Indicates the security level for secure invasive debug: `0b00` Functionality is not implemented or is controlled elsewhere.
[3:2]	NSNID	Indicates the security level for non-secure non-invasive debug: `0b00` Functionality is not implemented or is controlled elsewhere.
[1:0]	NSID	Indicates the security level for non-secure invasive debug: `0b00` Functionality is not implemented or is controlled elsewhere.

3.3.9 Device Architecture Register

The DEVARCH Register characteristics are:

Purpose Returns the device architecture identifier value.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-1 on page 3-3.

Figure 3-9 shows the bit assignments.

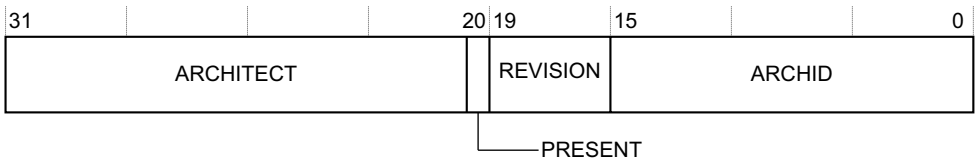


Figure 3-9 DEVARCH Register bit assignments

Table 3-10 shows the bit assignments.

Table 3-10 DEVARCH Register bit assignments

Bits	Name	Description
[31:21]	ARCHITECT	A value of `0x23B` indicates that ARM as the component architect.
[20]	PRESENT	A value of `0x1` indicates that the DEVARCH register is present.
[19:16]	REVISION	A value of `0x0` indicates that the revision is 0 for the architecture.
[15:0]	ARCHID	A value of `0x0A34` indicates that the component is a power requestor unit.

3.3.10 Device Configuration Register

The DEVID Register characteristics are:

Purpose Indicates the capabilities of the component.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-1 on page 3-3.

Figure 3-10 shows the bit assignments.

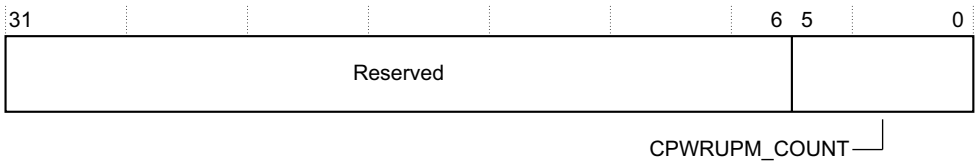


Figure 3-10 DEVID Register bit assignments

Table 3-11 shows the bit assignments.

Table 3-11 DEVID Register bit assignments

Bits	Name	Function
[31:6]	Reserved	-
[5:0]	CPWRUPM_COUNT	This value represents the number of CPWRUP master interfaces on the GPR component. Permitted range of values is between `0x1` and `0x20`.

3.3.11 Device Type Identifier Register

The DEVTYPE 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 See the register summary in Table 3-1 on page 3-3.

Figure 3-11 shows the bit assignments.

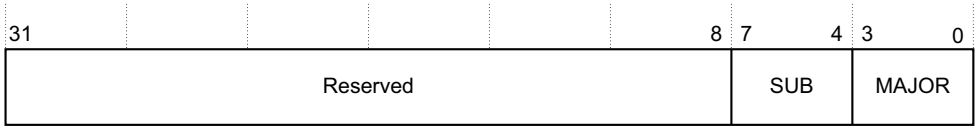


Figure 3-11 DEVTYPE Register bit assignments

Table 3-12 shows the bit assignments.

Table 3-12 DEVTYPE Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	SUB	Sub-classification of the type of the debug component as specified in the *CoreSight Architecture Specification* within the major classification as specified in the MAJOR field. `0b0011` Indicates that this component controls powering up and powering down the components in a CoreSight system.
[3:0]	MAJOR	Major classification of the type of the debug component as specified in the *CoreSight Architecture Specification* for this debug and trace component. `0b0100` Indicates that this component allows a debugger to control other components in a CoreSight system.

3.3.12 Peripheral ID4 Register

The PIDR4 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-1 on page 3-3.

Figure 3-12 shows the bit assignments.

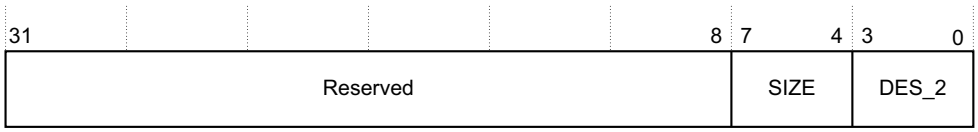


Figure 3-12 PIDR4 bit assignments

Table 3-13 shows the bit assignments.

Table 3-13 PIDR4 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	SIZE	This is a 4-bit value that indicates the total contiguous size of the memory window this component uses in powers of two, from the standard 4KB size. `0b0000` Indicates that the device only occupies 4KB of memory.
[3:0]	DES_2	This is the JEDEC JEP106 continuation code. This code, along with bits[6:4] of the identity code defined in the PIDR0 Register and bits[3:0] of the identity code defined in the PIDR1 Register, identifies the designer of the component. `0b0100` JEDEC continuation code.

3.3.13 Peripheral ID5-7 registers

The PIDR5-7 characteristics are:

Purpose Reserved.

Usage constraints There are no usage constraints.

Configurations These registers are available in all configurations.

Attributes See the register summary in Table 3-1 on page 3-3.

Figure 3-13 shows the PIDR5-7 bit assignments.

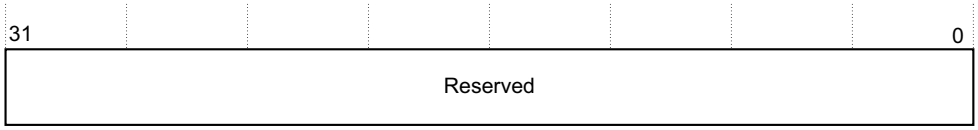


Figure 3-13 PIDR5-7 bit assignments

Table 3-14 shows the PIDR5-7 bit assignments.

Table 3-14 PIDR5-7 bit assignments

Bits	Name	Function
[31:0]	Reserved	-

3.3.14 Peripheral ID0 Register

The PIDR0 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 See the register summary in Table 3-1 on page 3-3.

Figure 3-14 on page 3-20 shows the bit assignments.

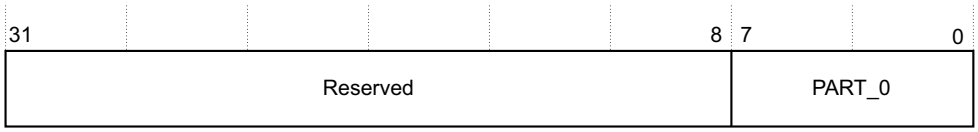


Figure 3-14 PIDR0 bit assignments

Table 3-15 shows the bit assignments.

Table 3-15 PIDR0 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PART_0	Bits[7:0] of the 12-bit part number of the component. The designer of the component assigns this part number. 0xA4 Indicates bits[7:0] of the part number of the component.

3.3.15 Peripheral ID1 Register

The PIDR1 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 See the register summary in Table 3-1 on page 3-3.

Figure 3-15 shows the bit assignments.

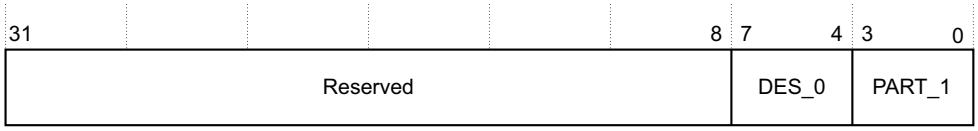


Figure 3-15 PIDR1 bit assignments

Table 3-16 shows the bit assignments.

Table 3-16 PIDR1 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	DES_0	Bits[3:0] of the JEDEC JEP106 identity code. This code, along with bits[6:4] of the identity code defined in the PIDR2 and the continuation code defined in the PIDR4, identifies the designer of the component. 0b1011 Bits[3:0] of the JEDEC JEP106 Identity Code. The default value is ARM.
[3:0]	PART_1	Bits[11:8] of the 12-bit part number of the component. The designer of the component assigns this part number. 0b1001 Indicates bits[11:8] of the part number of the component.

3.3.16 Peripheral ID2 Register

The PIDR2 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 See the register summary in Table 3-1 on page 3-3.

Figure 3-16 shows the bit assignments.

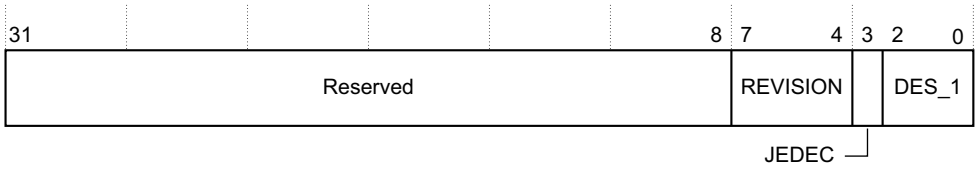


Figure 3-16 PIDR2 bit assignments

Table 3-17 shows the bit assignments.

Table 3-17 PIDR2 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	REVISION	An incremental value starting from `0b0000` for the first revision of this component. This increases by 1 for both major and minor revisions and is used to identify the major or minor revisions. `0b0000` This device is at r0p0.
[3]	JEDEC	Always set. Indicates whether the JEDEC assigned designer ID is used. `0b1` Indicates that a JEDEC assigned designer ID is used.
[2:0]	DES_1	Bits[6:4] of the JEDEC JEP106 identity code. This code, along with bits[3:0] of the identity code defined in the PIDR1 and the continuation code defined in the PIDR4, identifies the designer of the component. `0b011` Bits[6:4] of the JEDEC JEP106 Identity Code. The default value is ARM.

3.3.17 Peripheral ID3 Register

The PIDR3 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-1 on page 3-3.

Figure 3-17 on page 3-22 shows the bit assignments.

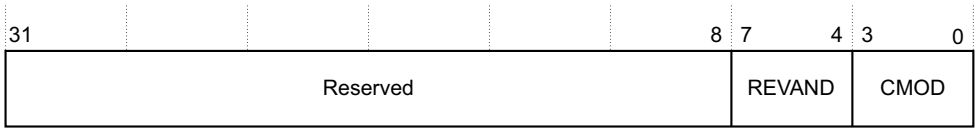


Figure 3-17 PIDR3 bit assignments

Table 3-18 shows the bit assignments.

Table 3-18 PIDR3 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	REVAND	Indicates minor errata fixes specific to the revision of the component being used, for example metal fixes after implementation. In most cases, this field is zero. ARM recommends that the component designers ensure that a metal fix can change this field if required, for example, by driving it from registers that reset to zero. `0b0000` Indicates that there are no errata fixes to this component.
[3:0]	CMOD	Indicates whether the customer has modified the behavior of the component. In most cases, this field is zero. The customer changes this value on modifications to this component. `0b0000` Indicates that the customer has not modified this component.

3.3.18 Component ID0 Register

The CIDR0 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-1 on page 3-3.

Figure 3-18 shows the bit assignments.

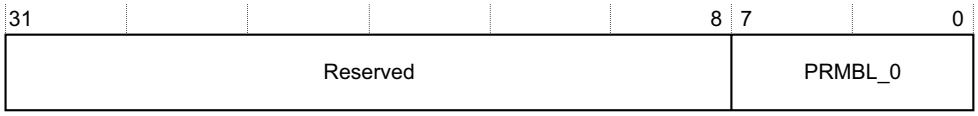


Figure 3-18 CIDR0 bit assignments

Table 3-19 shows the bit assignments.

Table 3-19 CIDR0 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_0	Contains bits[7:0] of the component identification code. `0x0D` Bits[7:0] of the identification code.

3.3.19 Component ID1 Register

The CIDR1 characteristics are:

Purpose A component identification register that indicates 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 See the register summary in Table 3-1 on page 3-3.

Figure 3-19 shows the bit assignments.

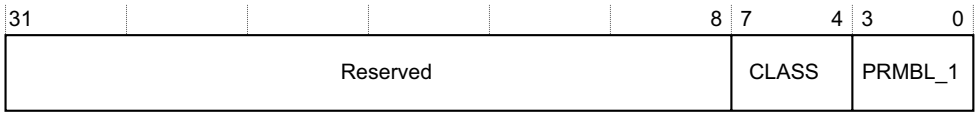


Figure 3-19 CIDR1 bit assignments

Table 3-20 shows the bit assignments.

Table 3-20 CIDR1 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	CLASS	Class of the component. Indicates, for example, whether the component is a ROM table or a generic CoreSight component. Contains bits[15:12] of the component identification code. `0b1001` Indicates that the component is a CoreSight component.
[3:0]	PRMBL_1	Contains bits[11:8] of the component identification code. `0b0000` Bits[11:8] of the identification code.

3.3.20 Component ID2 Register

The CIDR2 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-1 on page 3-3.

Figure 3-20 shows the bit assignments.

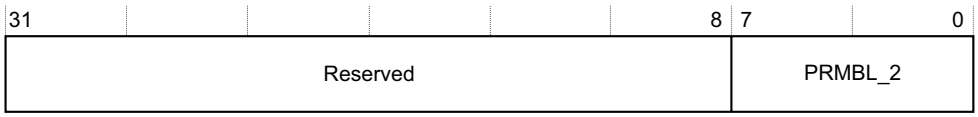


Figure 3-20 CIDR2 bit assignments

Table 3-21 shows the bit assignments.

Table 3-21 CIDR2 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_2	Contains bits[23:16] of the component identification code. `0x05` Bits[23:16] of the identification code.

3.3.21 Component ID3 Register

The CIDR3 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-1 on page 3-3.

Figure 3-21 shows the bit assignments.

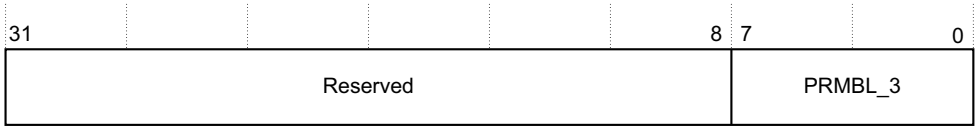


Figure 3-21 CIDR3 bit assignments

Table 3-22 shows the bit assignments.

Table 3-22 CIDR3 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_3	Contains bits[31:24] of the component identification code. `0xB1` Bits[31:24] of the identification code.

3.4 APB interconnect register summary

Table 3-23 shows the APB interconnect registers in offset order from the base memory address.

Table 3-23 APB interconnect register summary

Offset	Name	Type	Reset	Description
0x000-0x0FC	ROM_ENTRY_*n*[a]	RO	-[b]	*ROM Table Entry* on page 3-26
0xFD0	PIDR4	RO	UNKNOWN[c]	*Peripheral ID4 Register* on page 3-27
0xFD4	PIDR5	RO	0x00000000	*Peripheral ID5-7 registers* on page 3-27
0xFD8	PIDR6	RO	0x00000000	
0xFDC	PIDR7	RO	0x00000000	
0xFE0	PIDR0	RO	-[d]	*Peripheral ID0 Register* on page 3-28
0xFE4	PIDR1	RO	UNKNOWN[e]	*Peripheral ID1 Register* on page 3-29
0xFE8	PIDR2	RO	UNKNOWN[f]	*Peripheral ID2 Register* on page 3-29
0xFEC	PIDR3	RO	0x00000000	*Peripheral ID3 Register* on page 3-30
0xFF0	CIDR0	RO	0x0000000D	*Component ID0 Register* on page 3-31
0xFF4	CIDR1	RO	0x00000010	*Component ID1 Register* on page 3-31
0xFF8	CIDR2	RO	0x00000005	*Component ID2 Register* on page 3-32
0xFFC	CIDR3	RO	0x000000B1	*Component ID3 Register* on page 3-32

a. Where *n* is 0-63.

b. The reset value depends on the value of the MASTER_INTFn_BASE_ADDR parameter, where *n* is 0-63. See *Additional reading* on page ix.

c. See Table 3-25 on page 3-27 for more information on the reset value and its dependencies.

d. The reset value depends on the system configuration, and identifies this as either a generic ROM table or a top-level ROM table.

e. See Table 3-28 on page 3-29 for more information on the reset value and its dependencies.

f. See Table 3-29 on page 3-30 for more information on the reset value and its dependencies.

3.5 APB interconnect register descriptions

This section describes the APB interconnect registers. Table 3-23 on page 3-25 provides cross-references to individual registers.

3.5.1 ROM Table Entry

The ROM_ENTRY_*n* Register characteristics are:

Purpose Returns the value of ROM_ENTRY_*n*, where *n* is 0-63.

Usage constraints There are no usage constraints.

Configurations The SoC designer determines the availability of this register at the time of implementation.

Attributes See the register summary in Table 3-23 on page 3-25.

Figure 3-22 shows the bit assignments.

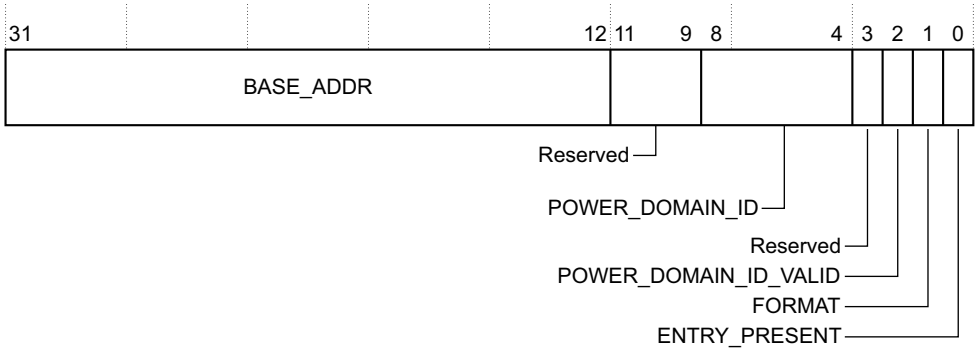


Figure 3-22 ROM_ENTRY_*n* Register bit assignments

Table 3-24 shows the bit assignments.

Table 3-24 ROM_ENTRY_*n* Register bit assignments

Bits	Name	Function
[31:12]	BASE_ADDR	Base address for master interface 0. Bit[31] is always zero.
[11:9]	Reserved	-
[8:4]	POWER_DOMAIN_ID	Indicates the power domain ID of the component. This field is only valid if bit[2] is `0b1`, otherwise this field must read as zero. Up to 32 power domains are supported using the values from `0x00` to `0x1F`.
[3]	Reserved	-

Table 3-24 ROM_ENTRY_*n* Register bit assignments (continued)

Bits	Name	Function
[2]	POWER_DOMAIN_ID_VALID	Indicates whether there is a power domain ID specified in the ROM Table entry. **0** Indicates that the POWER_DOMAIN_ID field of this register is not valid. **1** Indicates that the POWER_DOMAIN_ID field of this register is valid.
[1]	FORMAT	Indicates the ROM table entry format, for example, 32-bit or other formats. **1** ROM table entry is of 32-bit format.
[0]	ENTRY_PRESENT	Indicates whether there is a valid ROM entry at this location. **0** Valid ROM table entry not present at this address location. **1** Valid ROM table entry present at this address location.

3.5.2 Peripheral ID4 Register

The PIDR4 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-23 on page 3-25.

Figure 3-23 shows the bit assignments.

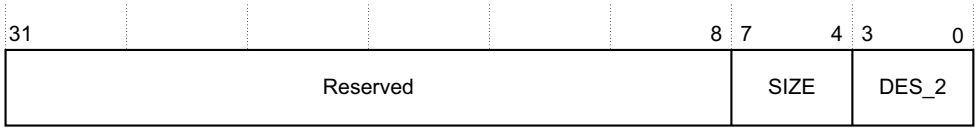


Figure 3-23 PIDR4 bit assignments

Table 3-25 shows the bit assignments.

Table 3-25 PIDR4 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	SIZE	This is a 4-bit value that indicates the total contiguous size of the memory window this component uses in powers of 2 from the standard 4KB. `0b0000` Indicates that the device only occupies 4KB of memory.
[3:0]	DES_2	This is the JEDEC JEP106 continuation code. This code, along with bits[6:4] of the identity code defined in the PIDR0, and with bits[3:0] of the identity code defined in the PIDR1, identifies the designer of the component. This reflects either **targetid[11:8]** from the DAP, or a sub-system specific value.

3.5.3 Peripheral ID5-7 registers

The PIDR5-7 characteristics are:

Purpose Reserved.

Usage constraints There are no usage constraints.

Configurations These registers are available in all configurations.

Attributes See the register summary in Table 3-23 on page 3-25.

Figure 3-24 shows the PIDR5-7 bit assignments.

Figure 3-24 PIDR5-7 bit assignments

Table 3-26 shows the PIDR5-7 bit assignments.

Table 3-26 PIDR5-7 bit assignments

Bits	Name	Function
[31:0]	Reserved	-

3.5.4 Peripheral ID0 Register

The PIDR0 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 See the register summary in Table 3-23 on page 3-25.

Figure 3-25 shows the bit assignments.

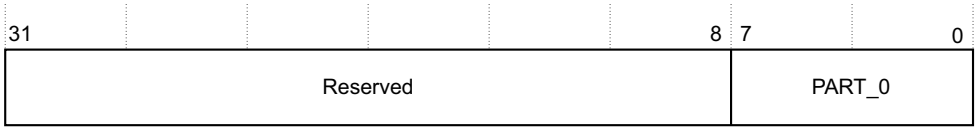


Figure 3-25 PIDR0 bit assignments

Table 3-27 shows the bit assignments.

Table 3-27 PIDR0 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PART_0	Bits[7:0] of the 12-bit part number of the component. The designer of the component assigns this part number. This reflects either **targetid[23:16]** from the DAP, or a sub-system identifier.

3.5.5 Peripheral ID1 Register

The PIDR1 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 See the register summary in Table 3-23 on page 3-25.

Figure 3-26 shows the bit assignments.

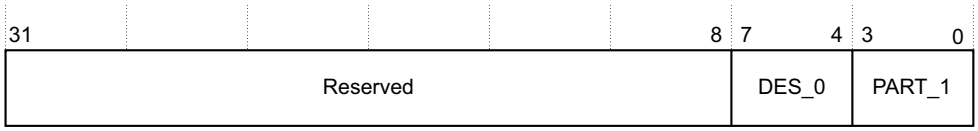


Figure 3-26 PIDR1 bit assignments

Table 3-28 shows the bit assignments.

Table 3-28 PIDR1 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	DES_0	Bits[3:0] of the JEDEC JEP106 identity code. This code, along with bits[6:4] of the identity code defined in the PIDR2 and the continuation code defined in the PIDR4, identifies the designer of the component. This reflects either the **targetid[4:1]** from the DAP, or a sub-system identifier.
[3:0]	PART_1	Bits[11:8] of the 12-bit part number of the component. The designer of the component assigns this part number. This reflects either the **targetid[27:24]** from the DAP, or a sub-system identifier.

3.5.6 Peripheral ID2 Register

The PIDR2 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 See the register summary in Table 3-23 on page 3-25.

Figure 3-27 shows the bit assignments.

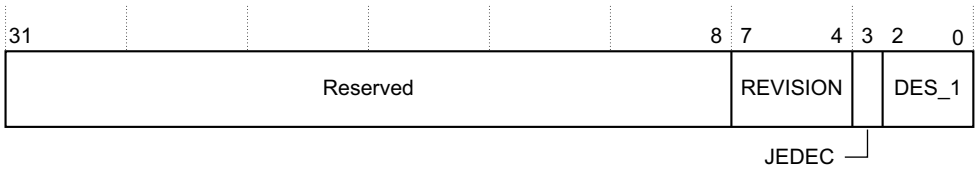


Figure 3-27 PIDR2 bit assignments

Table 3-29 shows the bit assignments.

Table 3-29 PIDR2 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	REVISION	An incremental value starting from `0b0000` for the first revision of this component. This increases by 1 for both major and minor revisions and is used to identify the major or minor revisions. This reflects either the **targetid[31:28]** from the DAP, or a sub-system identifier.
[3]	JEDEC	Always set. Indicates whether the JEDEC assigned designer ID is used. `0b1` Indicates that a JEDEC assigned designer ID is used.
[2:0]	DES_1	Bits[6:4] of the JEDEC JEP106 identity code. This code, along with bits[3:0] of the identity code defined in the PIDR1 and the continuation code defined in the PIDR4, identifies the designer of the component. This reflects either the **targetid[7:5]** from the DAP, or a sub-system identifier.

3.5.7 Peripheral ID3 Register

The PIDR3 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-23 on page 3-25.

Figure 3-28 shows the bit assignments.

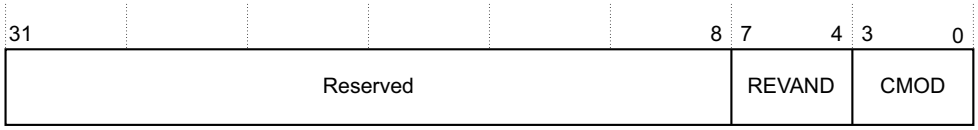


Figure 3-28 PIDR3 bit assignments

Table 3-30 shows the bit assignments.

Table 3-30 PIDR3 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	REVAND	Indicates minor errata fixes specific to the revision of the component being used, for example metal fixes after implementation. In most cases, this field is zero. ARM recommends that the component designers ensure that a metal fix can change this field, if required, for example, by driving it from registers that reset to zero. `0b0000` Indicates that there are no errata fixes to this component.
[3:0]	CMOD	Indicates whether the customer has modified the behavior of the component. In most cases, this field is zero. The customer changes this value on modifications to this component. `0b0000` Indicates that the customer has not modified this component.

3.5.8 Component ID0 Register

The CIDR0 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-23 on page 3-25.

Figure 3-29 shows the bit assignments.

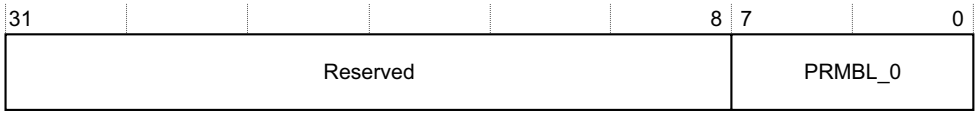


Figure 3-29 CIDR0 bit assignments

Table 3-31 shows the bit assignments.

Table 3-31 CIDR0 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_0	Contains bits[7:0] of the component identification code. `0x0D` Bits[7:0] of the identification code.

3.5.9 Component ID1 Register

The CIDR1 characteristics are:

Purpose A component identification register that indicates 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 See the register summary in Table 3-23 on page 3-25.

Figure 3-30 shows the bit assignments.

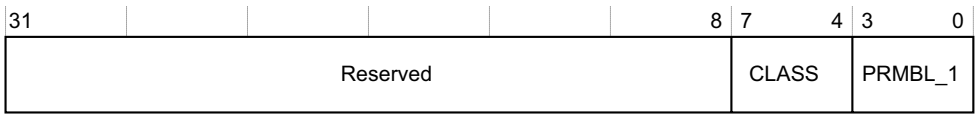


Figure 3-30 CIDR1 bit assignments

Table 3-32 shows the bit assignments.

Table 3-32 CIDR1 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	CLASS	Class of the component, for example, if the component is a ROM table or a generic CoreSight component. Contains bits[15:12] of the component identification code. `0b0001` Indicates that the component is a ROM table.
[3:0]	PRMBL_1	Contains bits[11:8] of the component identification code. `0b0000` Bits[11:8] of the identification code.

3.5.10 Component ID2 Register

The CIDR2 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-23 on page 3-25.

Figure 3-31 shows the bit assignments.

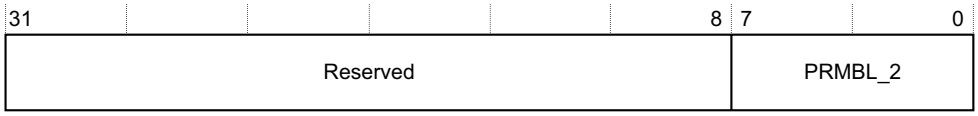


Figure 3-31 CIDR2 bit assignments

Table 3-33 shows the bit assignments.

Table 3-33 CIDR2 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_2	Contains bits[23:16] of the component identification code. `0x05` Bits[23:16] of the identification code.

3.5.11 Component ID3 Register

The CIDR3 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-23 on page 3-25.

Figure 3-32 shows the bit assignments.

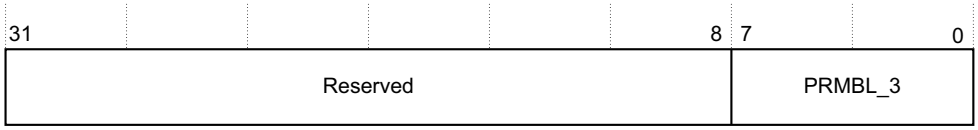


Figure 3-32 CIDR3 bit assignments

Table 3-34 shows the bit assignments.

Table 3-34 CIDR3 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_3	Contains bits[31:24] of the component identification code. `0xB1` Bits[31:24] of the identification code.

3.6 ATB funnel register summary

Table 3-35 shows the ATB funnel registers in offset order from the base memory address.

Table 3-35 ATB funnel register summary

Offset	Name	Type	Reset	Description
0x000	Ctrl_Reg	RW	0x00000300	*Funnel Control Register* on page 3-35
0x004	Priority_Ctrl_Reg	RW	0x00000000	*Priority Control Register* on page 3-37
0xEEC	ITATBDATA0	RW	0x00000000	*Integration Test ATB Data0 Register* on page 3-40
0xEF0	ITATBCTR2	RW	0x00000000	*Integration Test ATB Control 2 Register* on page 3-43
0xEF4	ITATBCTR1	RW	0x00000000	*Integration Test ATB Control 1 Register* on page 3-44
0xEF8	ITATBCTR0	RW	0x00000000	*Integration Test ATB Control 0 Register* on page 3-45
0xF00	ITCTRL	RW	0x00000000	*Integration Mode Control Register* on page 3-45
0xFA0	CLAIMSET	RW	0x0000000F	*Claim Tag Set Register* on page 3-46
0xFA4	CLAIMCLR	RW	0x00000000	*Claim Tag Clear Register* on page 3-47
0xFB0	LOCKACCESS	WO	0x00000000	*Lock Access Register* on page 3-47
0xFB4	LOCKSTATUS	RO	0x00000003	*Lock Status Register* on page 3-48
0xFB8	AUTHSTATUS	RO	0x00000000	*Authentication Status Register* on page 3-49
0xFC8	DEVID	RO	0x00000038	*Device Configuration Register* on page 3-50
0xFCC	DEVTYPE	RO	0x00000012	*Device Type Identifier Register* on page 3-51
0xFD0	PIDR4	RO	0x00000004	*Peripheral ID4 Register* on page 3-51
0xFD4	PIDR5	RO	0x00000000	*Peripheral ID5-7 registers* on page 3-52
0xFD8	PIDR6	RO	0x00000000	
0xFDC	PIDR7	RO	0x00000000	
0xFE0	PIDR0	RO	0x00000008	*Peripheral ID0 Register* on page 3-52
0xFE4	PIDR1	RO	0x000000B9	*Peripheral ID1 Register* on page 3-53
0xFE8	PIDR2	RO	0x0000002B	*Peripheral ID2 Register* on page 3-54
0xFEC	PIDR3	RO	0x00000000	*Peripheral ID3 Register* on page 3-54
0xFF0	CIDR0	RO	0x0000000D	*Component ID0 Register* on page 3-55
0xFF4	CIDR1	RO	0x00000090	*Component ID1 Register* on page 3-56
0xFF8	CIDR2	RO	0x00000005	*Component ID2 Register* on page 3-56
0xFFC	CIDR3	RO	0x000000B1	*Component ID3 Register* on page 3-57

3.7 ATB funnel register descriptions

This section describes the ATB funnel registers. Table 3-35 on page 3-34 provides cross-references to individual registers.

The registers are only present if the APB programming interface is chosen.

3.7.1 Funnel Control Register

The Ctrl_Reg Register characteristics are:

Purpose The Funnel Control Register enables the slave ports and defines the hold time of the slave ports. Hold time refers to the number of transactions that are output on the funnel master port from the same slave when that slave port **atvalidsx** is HIGH. Hold time does not mention clock cycles in this context.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

The number of bits in this register depends on the number of slaves selected in the configuration.

Attributes See the register summary in Table 3-35 on page 3-34.

Figure 3-33 shows the bit assignments.

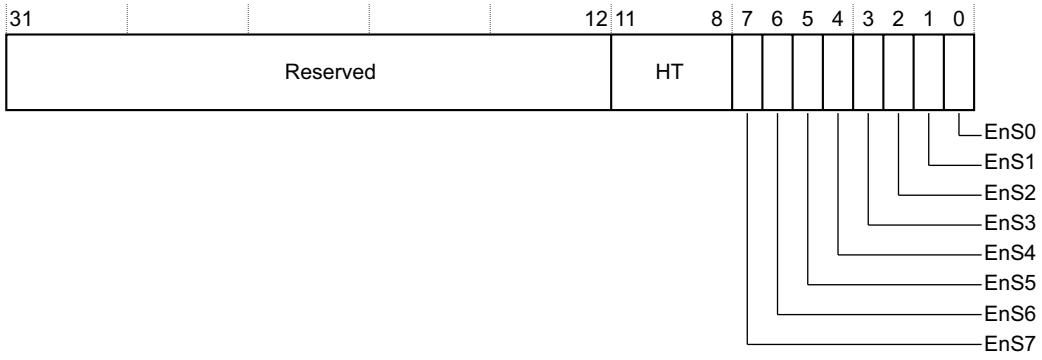


Figure 3-33 Ctrl_Reg Register bit assignments

Table 3-36 shows the bit assignments.

Table 3-36 Ctrl_Reg Register bit assignments

Bits	Name	Function
[31:12]	Reserved	-
[11:8]	HT	The formatting scheme can become inefficient if fast switching occurs, so where possible, minimize this. If a source has nothing to transmit, then another source is selected irrespective of the minimum number of transactions. The ATB funnel holds for the minimum hold time and one additional transaction. The actual hold time is the register value plus 1. The maximum value that can be entered is `0b1110` and this equates to 15 transactions. `0b1111` is reserved. `0b0000` 1 transaction hold time. `0b0001` 2 transactions hold time. `0b0010` 3 transactions hold time. `0b0011` 4 transactions hold time. `0b0100` 5 transactions hold time. `0b0101` 6 transactions hold time. `0b0110` 7 transactions hold time. `0b0111` 8 transactions hold time. `0b1000` 9 transactions hold time. `0b1001` 10 transactions hold time. `0b1010` 11 transactions hold time. `0b1011` 12 transactions hold time. `0b1100` 13 transactions hold time. `0b1101` 14 transactions hold time. `0b1110` 15 transactions hold time.
[7]	EnS7	Setting this bit enables this input or slave port. If the bit is not set, then this has the effect of excluding the port from the priority selection scheme. The reset value is zero. **0** Slave port disabled. **1** Slave port enabled.
[6]	EnS6	Setting this bit enables this input or slave port. If the bit is not set, then this has the effect of excluding the port from the priority selection scheme. The reset value is zero. **0** Slave port disabled. **1** Slave port enabled.
[5]	EnS5	Setting this bit enables this input or slave port. If the bit is not set, then this has the effect of excluding the port from the priority selection scheme. The reset value is zero. **0** Slave port disabled. **1** Slave port enabled.
[4]	EnS4	Setting this bit enables this input or slave port. If the bit is not set, then this has the effect of excluding the port from the priority selection scheme. The reset value is zero. **0** Slave port disabled. **1** Slave port enabled.
[3]	EnS3	Setting this bit enables this input or slave port. If the bit is not set, then this has the effect of excluding the port from the priority selection scheme. The reset value is zero. **0** Slave port disabled. **1** Slave port enabled.

Table 3-36 Ctrl_Reg Register bit assignments (continued)

Bits	Name	Function
[2]	EnS2	Setting this bit enables this input or slave port. If the bit is not set, then this has the effect of excluding the port from the priority selection scheme. The reset value zero. **0** Slave port disabled. **1** Slave port enabled.
[1]	EnS1	Setting this bit enables this input or slave port. If the bit is not set, then this has the effect of excluding the port from the priority selection scheme. The reset value zero. **0** Slave port disabled. **1** Slave port enabled.
[0]	EnS0	Setting this bit enables this input or slave port. If the bit is not set, then this has the effect of excluding the port from the priority selection scheme. The reset value zero. **0** Slave port disabled. **1** Slave port enabled.

3.7.2 Priority Control Register

The Priority_Ctrl_Reg Register characteristics are:

Purpose The Priority Control Register defines the order in which inputs are selected. Each 3-bit field represents a priority for each particular slave interface. For example:

- **Location 0** Has the priority value for the first slave port.
- **Location 1** Has the priority value for the second slave port.
- **Location 2** Has the priority value for the third slave port.
- **...** ...
- **...** ...
- **Location 7** Has the priority value for the eighth slave port.

The values represent the priority value for each port number. If you want to give the highest priority to a particular slave port, program the corresponding port with the lowest value. Typically, this is likely to be a port that has more important data or that has a small FIFO and is therefore likely to overflow. If you want to give lowest priority to a particular slave port, program the corresponding slave port with the highest value.

Usage constraints There are no usage constraints. Priority values cannot be changed while the corresponding slave port is enabled.

Configurations This register is available in all configurations.

The number of bits in this register depends on the number of slaves selected in the configuration.

Attributes See the register summary in Table 3-35 on page 3-34.

Figure 3-34 on page 3-38 shows the bit assignments.

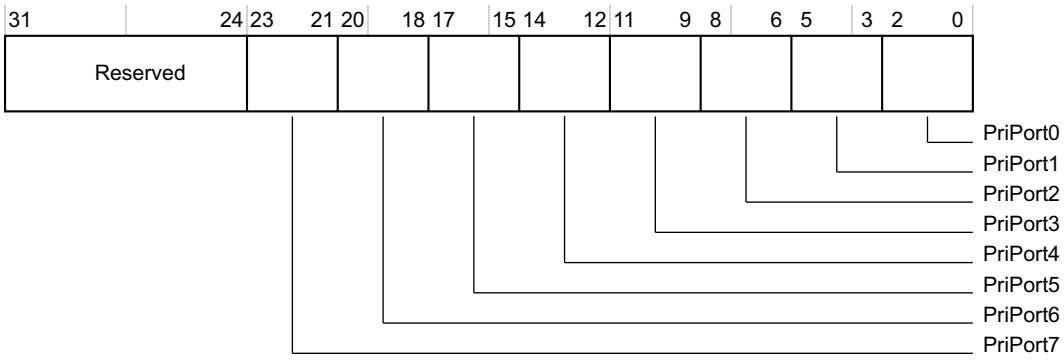


Figure 3-34 Priority_Ctrl_Reg Register bit assignments

Table 3-37 shows the bit assignments.

Table 3-37 Priority_Ctrl_Reg Register bit assignments

Bits	Name	Function
[31:24]	Reserved	-
[23:21]	PriPort7	Priority value of the eighth port. The value written into this location is the value that you want to assign the eighth slave port. The arbitration logic selects the slave with the lowest port number when the same priority value is entered for multiple slaves. `0b000` Represents the priority value of 0. `0b001` Represents the priority value of 1. `0b010` Represents the priority value of 2. `0b011` Represents the priority value of 3. `0b100` Represents the priority value of 4. `0b101` Represents the priority value of 5. `0b110` Represents the priority value of 6. `0b111` Represents the priority value of 7.
[20:18]	PriPort6	Priority value of the seventh port. The value written into this location is the value that you want to assign the seventh slave port. The arbitration logic selects the slave with the lowest port number when the same priority value is entered for multiple slaves. `0b000` Represents the priority value of 0. `0b001` Represents the priority value of 1. `0b010` Represents the priority value of 2. `0b011` Represents the priority value of 3. `0b100` Represents the priority value of 4. `0b101` Represents the priority value of 5. `0b110` Represents the priority value of 6. `0b111` Represents the priority value of 7.

Table 3-37 Priority_Ctrl_Reg Register bit assignments (continued)

Bits	Name	Function
[17:15]	PriPort5	Priority value of the sixth port. The value written into this location is the value that you want to assign the sixth slave port. The arbitration logic selects the slave with the lowest port number when the same priority value is entered for multiple slaves. 0b000 Represents the priority value of 0. 0b001 Represents the priority value of 1. 0b010 Represents the priority value of 2. 0b011 Represents the priority value of 3. 0b100 Represents the priority value of 4. 0b101 Represents the priority value of 5. 0b110 Represents the priority value of 6. 0b111 Represents the priority value of 7.
[14:12]	PriPort4	Priority value of the fifth port. The value written into this location is the value that you want to assign the fifth slave port. The arbitration logic selects the slave with the lowest port number when the same priority value is entered for multiple slaves. 0b000 Represents the priority value of 0. 0b001 Represents the priority value of 1. 0b010 Represents the priority value of 2. 0b011 Represents the priority value of 3. 0b100 Represents the priority value of 4. 0b101 Represents the priority value of 5. 0b110 Represents the priority value of 6. 0b111 Represents the priority value of 7.
[11:9]	PriPort3	Priority value of the fourth port.The value written into this location is the value that you want to assign the fourth slave port. The arbitration logic selects the slave with the lowest port number when the same priority value is entered for multiple slaves. 0b000 Represents the priority value of 0. 0b001 Represents the priority value of 1. 0b010 Represents the priority value of 2. 0b011 Represents the priority value of 3. 0b100 Represents the priority value of 4. 0b101 Represents the priority value of 5. 0b110 Represents the priority value of 6. 0b111 Represents the priority value of 7.

Table 3-37 Priority_Ctrl_Reg Register bit assignments (continued)

Bits	Name	Function
[8:6]	PriPort2	Priority value of the third port. The value written into this location is the value that you want to assign the third slave port. The arbitration logic selects the slave with the lowest port number when the same priority value is entered for multiple slaves. 0b000 Represents the priority value of 0. 0b001 Represents the priority value of 1. 0b010 Represents the priority value of 2. 0b011 Represents the priority value of 3. 0b100 Represents the priority value of 4. 0b101 Represents the priority value of 5. 0b110 Represents the priority value of 6. 0b111 Represents the priority value of 7.
[5:3]	PriPort1	Priority value of the second port. The value written into this location is the value that you want to assign the second slave port. The arbitration logic selects the slave with the lowest port number when the same priority value is entered for multiple slaves. 0b000 Represents the priority value of 0. 0b001 Represents the priority value of 1. 0b010 Represents the priority value of 2. 0b011 Represents the priority value of 3. 0b100 Represents the priority value of 4. 0b101 Represents the priority value of 5. 0b110 Represents the priority value of 6. 0b111 Represents the priority value of 7.
[2:0]	PriPort0	Priority value of the first slave port. The value written into this location is the value that you want to assign the first slave port. The arbitration logic selects the slave with the lowest port number when the same priority value is entered for multiple slaves. 0b000 Represents the priority value of 0. 0b001 Represents the priority value of 1. 0b010 Represents the priority value of 2. 0b011 Represents the priority value of 3. 0b100 Represents the priority value of 4. 0b101 Represents the priority value of 5. 0b110 Represents the priority value of 6. 0b111 Represents the priority value of 7.

3.7.3 Integration Test ATB Data0 Register

The ITATBDATA0 Register characteristics are:

Purpose Performs different functions depending on whether the access is a read or a write:

- A read returns the data from **ATDATAS***n*, where *n* is the index of the slave port that is enabled. For information about ports that are enabled, see *Funnel Control Register* on page 3-35. The read data is only valid when **ATVALIDS***n* is HIGH.
- A write outputs data to the byte boundaries of **ATDATAM**.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-35 on page 3-34.

Figure 3-35 shows the bit assignments.

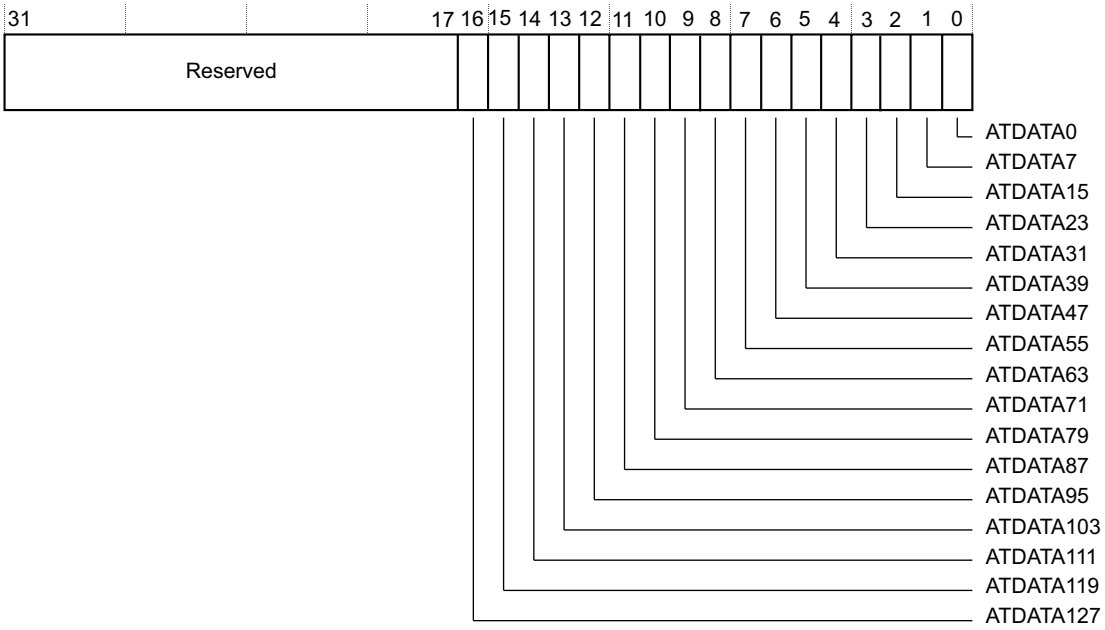


Figure 3-35 ITATBDATA0 Register bit assignments

Table 3-38 shows the bit assignments.

Table 3-38 ITATBDATA0 Register bit assignments

Bits	Name	Function
[31:17]	Reserved	-
[16]	ATDATA127	A read access returns the value of **ATDATAS<x>[127]** of the enabled port. A write access writes to **ATDATAM[127]** of the enabled port. **0** **ATDATA[127]** of the enabled port is LOW. **1** **ATDATA[127]** of the enabled port is HIGH.
[15]	ATDATA119	A read access returns the value of **ATDATAS<x>[119]** of the enabled port. A write access writes to **ATDATAM[119]** of the enabled port. **0** **ATDATA[119]** of the enabled port is LOW. **1** **ATDATA[119]** of the enabled port is HIGH.
[14]	ATDATA111	A read access returns the value of **ATDATAS<x>[111]** of the enabled port. A write access writes to **ATDATAM[111]** of the enabled port. **0** **ATDATA[111]** of the enabled port is LOW. **1** **ATDATA[111]** of the enabled port is HIGH.
[13]	ATDATA103	A read access returns the value of **ATDATAS<x>[103]** of the enabled port. A write access writes to **ATDATAM[103]** of the enabled port. **0** **ATDATA[103]** of the enabled port is LOW. **1** **ATDATA[103]** of the enabled port is HIGH.

Table 3-38 ITATBDATA0 Register bit assignments (continued)

Bits	Name	Function
[12]	ATDATA95	A read access returns the value of **ATDATAS<x>[95]** of the enabled port. A write access writes to **ATDATAM[95]** of the enabled port. **0** **ATDATA[95]** of the enabled port is LOW. **1** **ATDATA[95]** of the enabled port is HIGH.
[11]	ATDATA87	A read access returns the value of **ATDATAS<x>[87]** of the enabled port. A write access writes to **ATDATAM[87]** of the enabled port. **0** **ATDATA[87]** of the enabled port is LOW. **1** **ATDATA[87]** of the enabled port is HIGH.
[10]	ATDATA79	A read access returns the value of **ATDATAS<x>[79]** of the enabled port. A write access writes to **ATDATAM[79]** of the enabled port. **0** **ATDATA[79]** of the enabled port is LOW. **1** **ATDATA[79]** of the enabled port is HIGH.
[9]	ATDATA71	A read access returns the value of **ATDATAS<x>[71]** of the enabled port. A write access writes to **ATDATAM[71]** of the enabled port. **0** **ATDATA[71]** of the enabled port is LOW. **1** **ATDATA[71]** of the enabled port is HIGH.
[8]	ATDATA63	A read access returns the value of **ATDATAS<x>[63]** of the enabled port. A write access writes to **ATDATAM[63]** of the enabled port. **0** **ATDATA[63]** of the enabled port is LOW. **1** **ATDATA[63]** of the enabled port is HIGH.
[7]	ATDATA55	A read access returns the value of **ATDATAS<x>[55]** of the enabled port. A write access writes to **ATDATAM[55] of the enabled port**. **0** **ATDATA[55]** of the enabled port is LOW. **1** **ATDATA[55]** of the enabled port is HIGH.
[6]	ATDATA47	A read access returns the value of **ATDATAS<x>[47]** of the enabled port. A write access writes to **ATDATAM[47]** of the enabled port. **0** **ATDATA[47]** of the enabled port is LOW. **1** **ATDATA[47]** of the enabled port is HIGH.
[5]	ATDATA39	A read access returns the value of **ATDATAS<x>[39]** of the enabled port. A write access writes to **ATDATAM[39]** of the enabled port. **0** **ATDATA[39]** of the enabled port is LOW. **1** **ATDATA[39]** of the enabled port is HIGH.
[4]	ATDATA31	A read access returns the value of **ATDATAS<x>[31]** of the enabled port. A write access writes to **ATDATAM[31]** of the enabled port. **0** **ATDATA[31]** of the enabled port is LOW. **1** **ATDATA[31]** of the enabled port is HIGH.
[3]	ATDATA23	A read access returns the value of **ATDATAS<x>[23]** of the enabled port. A write access writes to **ATDATAM[23]** of the enabled port. **0** **ATDATA[23]** of the enabled port is LOW. **1** **ATDATA[23]** of the enabled port is HIGH.

Table 3-38 ITATBDATA0 Register bit assignments (continued)

Bits	Name	Function
[2]	ATDATA15	A read access returns the value of **ATDATAS<x>[15]** of the enabled port. A write access writes to **ATDATAM[15]** of the enabled port. **0** **ATDATA[15]** of the enabled port is LOW. **1** **ATDATA[15]** of the enabled port is HIGH.
[1]	ATDATA7	A read access returns the value of **ATDATAS<x>[7]** of the enabled port. A write access writes to **ATDATAM[7]** of the enabled port. **0** **ATDATA[7]** of the enabled port is LOW. **1** **ATDATA[7]** of the enabled port is HIGH.
[0]	ATDATA0	A read access returns the value of **ATDATAS<x>[0]** of the enabled port. A write access writes to **ATDATAM[0]** of the enabled port. **0** **ATDATA[0]** of the enabled port is LOW. **1** **ATDATA[0]** of the enabled port is HIGH.

3.7.4 Integration Test ATB Control 2 Register

The ITATBCTR2 Register characteristics are:

Purpose Performs different functions depending on whether the access is a read or a write:

- A read returns the data from **atreadym** and **afvalidm**.
- A write outputs data on **atreadys***n* and **afvalids***n*, where the value of the ATB Funnel Control Register at `0x000` defines *n*.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-35 on page 3-34.

Figure 3-36 shows the bit assignments.

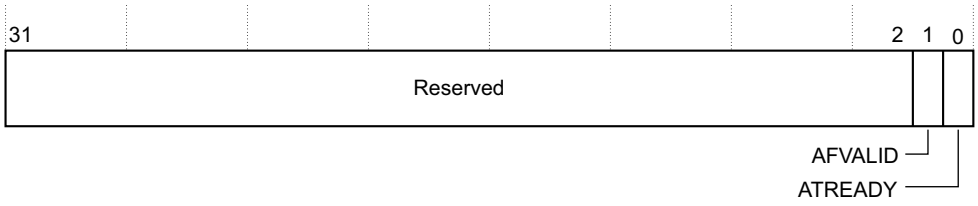


Figure 3-36 ITATBCTR2 Register bit assignments

Table 3-39 shows the bit assignments.

Table 3-39 ITATBCTR2 Register bit assignments

Bits	Name	Function
[31:2]	Reserved	-
[1]	AFVALID	A read access returns the value of **afvalidm**. A write access outputs the data to **afvalids***n*, where the value of the ATB Funnel Control Register at `0x000` defines n. **0** Pin is at logic 0. **1** Pin is at logic 1.
[0]	ATREADY	A read access returns the value of **atreadym**. A write access outputs the data to **atreadys***n*, where the value of the ATB Funnel Control Register at `0x000` defines n. **0** Pin is at logic 0. **1** Pin is at logic 1.

3.7.5 Integration Test ATB Control 1 Register

The ITATBCTR1 Register characteristics are:

Purpose Performs different functions depending on whether the access is a read or a write:

- A read returns the value of the **atids***n* signals, where the value of the Control Register at `0x000` defines *n*.
- A write operation in the register outputs the value written in the register to **atidm**.

The read data is only valid when **ATVALIDS***n* is HIGH.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-35 on page 3-34.

Figure 3-37 shows the bit assignments.

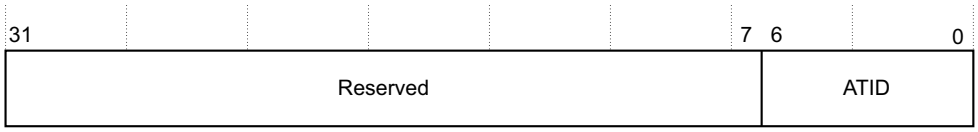


Figure 3-37 ITATBCTR1 Register bit assignments

Table 3-40 shows the bit assignments.

Table 3-40 ITATBCTR1 Register bit assignments

Bits	Name	Function
[31:7]	Reserved	-
[6:0]	ATID	A read returns the value of the **atids***n* signals, where the value of the Control Register at `0x000` defines *n*. A write outputs the value to the **atidm** port.

3.7.6 Integration Test ATB Control 0 Register

The ITATBCTR0 Register characteristics are:

Purpose Performs different functions depending on whether the access is a read or a write:

- A read returns the value of the **atvalids***n*, **atbytess***n*, and **afreadys***n* signals, where the value of the Control Register at `0x000` defines *n*.
- A write sets the value of the **atvalidm**, **atbytesm**, and **afreadym** signals.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-35 on page 3-34.

Figure 3-38 shows the bit assignments.

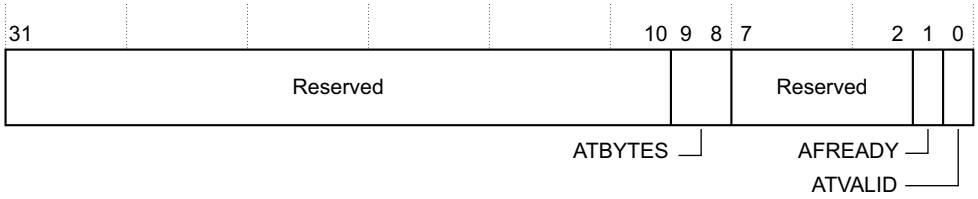


Figure 3-38 ITATBCTR0 Register bit assignments

Table 3-41 shows the bit assignments.

Table 3-41 ITATBCTR0 Register bit assignments

Bits	Name	Function
[31:10]	Reserved	-
[9:8]	ATBYTES	A read returns the value of the **atbytess***n* signal, where the value of the ATB Funnel Control Register at `0x000` defines *n*. A write outputs the value to **atbytesm**.
[7:2]	Reserved	-
[1]	AFREADY	A read returns the value of the **afreadys***n* signal, where the value of the ATB Funnel Control Register at `0x000` defines *n*. A write outputs the value to **afreadym**.
[0]	ATVALID	A read returns the value of the **atvalids***n* signal, where the value of the ATB Funnel Control Register at `0x000` defines *n*. A write outputs the value to **atvalidm**.

3.7.7 Integration Mode Control Register

The ITCTRL Register characteristics are:

Purpose Enables topology detection. For more information, see the *CoreSight Architecture Specification*. 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 the purposes of integration testing and topology detection.

Note

When a device is in integration mode, the intended functionality might not be available.

After performing integration or topology detection, reset the system to ensure correct behavior of CoreSight and other connected system components that the integration or topology detection can affect.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-35 on page 3-34.

Figure 3-39 shows the bit assignments.

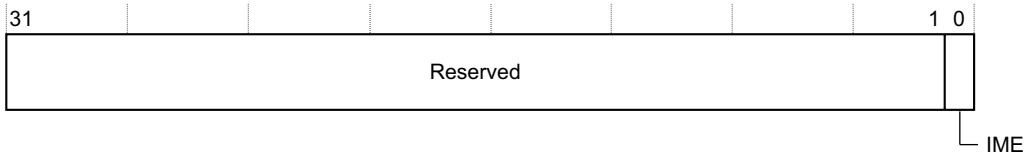


Figure 3-39 ITCTRL Register bit assignments

Table 3-42 shows the bit assignments.

Table 3-42 ITCTRL Register bit assignments

Bits	Name	Function
[31:1]	Reserved	-
[0]	IME	Enables the component to switch between functional mode and integration mode. If no integration functionality is implemented, this register must read as zero. **0** Disable integration mode. **1** Enable integration mode.

3.7.8 Claim Tag Set Register

The CLAIMSET Register characteristics are:

Purpose Software can use the claim tag to coordinate application and debugger access to trace unit functionality. The claim tags have no effect on the operation of the component. The CLAIMSET Register sets bits in the claim tag, and determines the number of claim bits implemented.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-35 on page 3-34.

Figure 3-40 shows the bit assignments.

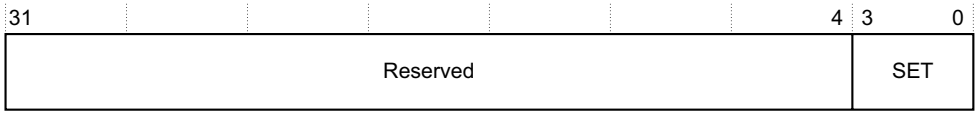


Figure 3-40 CLAIMSET Register bit assignments

Table 3-43 shows the bit assignments.

Table 3-43 CLAIMSET Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	SET_W	When you write 0 to bit[n] of this register, there is no effect on the claim tag bit[n]. When you write 1 to bit[n] of this register, bit[n] in the claim tag is set to 1.
[3:0]	SET_R	When you read 0 on a particular bit of this register, it indicates that this claim tag bit is not implemented. When you read 1 on a particular bit of this register, it indicates that this claim tag bit is implemented. `0b1111` Indicates that four bits of claim tag are implemented.

3.7.9 Claim Tag Clear Register

The CLAIMCLR Register characteristics are:

Purpose Software can use the claim tag to coordinate application and debugger access to trace unit functionality. The claim tags have no effect on the operation of the component. The CLAIMCLR Register clears the bits in the claim tag, and determines the current value of the claim tag.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-35 on page 3-34.

Figure 3-41 shows the bit assignments.

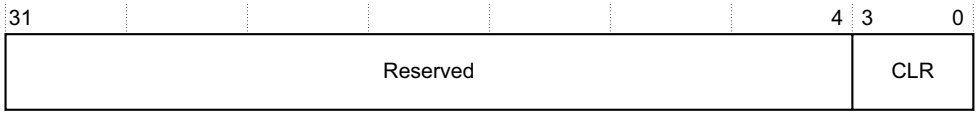


Figure 3-41 CLAIMCLR Register bit assignments

Table 3-44 shows the bit assignments.

Table 3-44 CLAIMCLR Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	CLR_W	When you write 0 to bit[n] of this register, there is no effect on the claim tag bit[n]. When you write 1 to bit[n] of this register, bit[n] in the claim tag is cleared to 0.
[3:0]	CLR_R	The value present reflects the present value of the Claim Tag.

3.7.10 Lock Access Register

The LAR characteristics are:

Purpose Controls write access from self-hosted, on-chip accesses. The LAR does not affect the accesses using the external debugger interface.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-35 on page 3-34.

Figure 3-42 shows the bit assignments.

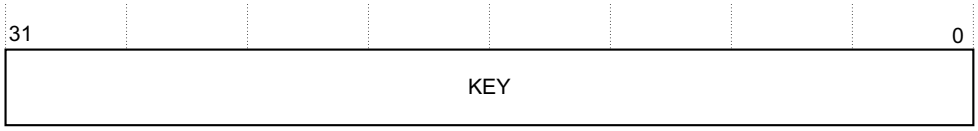


Figure 3-42 LAR bit assignments

Table 3-45 shows the bit assignments.

Table 3-45 LAR bit assignments

Bits	Name	Function
[31:0]	KEY	When you write `0xC5ACCE55`, subsequent write operations to this device are enabled. Any other value disables subsequent write operations.

3.7.11 Lock Status Register

The LSR characteristics are:

Purpose Indicates the status of the lock control mechanism. This lock prevents accidental writes by code. When locked, write accesses are denied for all registers except for the LAR. The lock registers do not affect accesses from the external debug interface. This register reads as 0 when accessed from the external debug interface.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-35 on page 3-34.

Figure 3-43 shows the bit assignments.

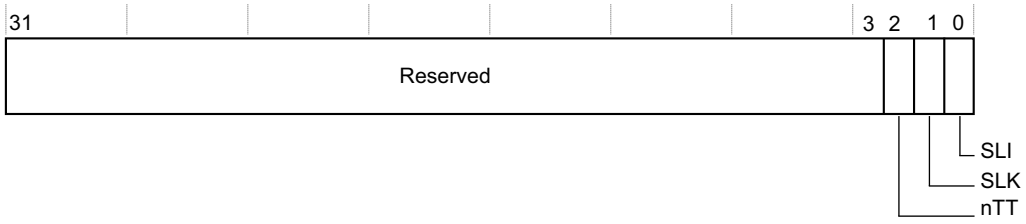


Figure 3-43 LSR bit assignments

Table 3-46 shows the bit assignments.

Table 3-46 LSR bit assignments

Bits	Name	Function
[31:3]	Reserved	-
[2]	nTT	Indicates that the LAR is implemented as 32-bit. `0b0` This component implements a 32-bit LAR.
[1]	SLK	Returns the present lock status of the device. `0b0` Indicates that write operations are permitted in this device. `0b1` Indicates that write operations are not permitted in this device. Read operations are permitted.
[0]	SLI	Indicates that a lock control mechanism is present in this device. `0b0` Indicates that a lock control mechanism is not present in this device. Write operations to the LAR are ignored. `0b1` Indicates that a lock control mechanism is present in this device.

3.7.12 Authentication Status Register

The AUTHSTATUS Register characteristics are:

Purpose Reports the required security level and present status.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-35 on page 3-34.

Figure 3-44 shows the bit assignments.

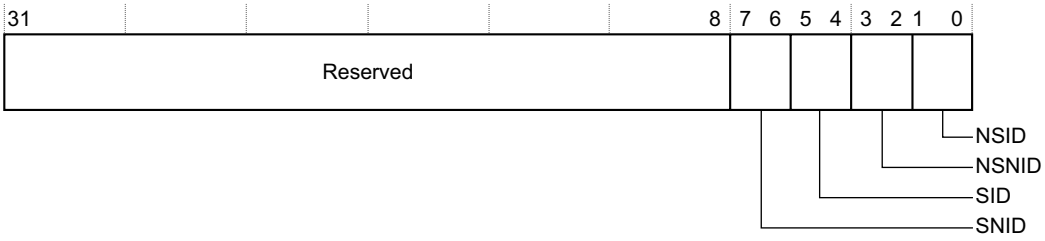


Figure 3-44 AUTHSTATUS Register bit assignments

Table 3-47 shows the bit assignments.

Table 3-47 AUTHSTATUS Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:6]	SNID	Indicates the security level for secure non-invasive debug: `0b00` Functionality is not implemented or is controlled elsewhere.

Table 3-47 AUTHSTATUS Register bit assignments (continued)

Bits	Name	Function
[5:4]	SID	Indicates the security level for secure invasive debug: 0b00 Functionality is not implemented or is controlled elsewhere.
[3:2]	NSNID	Indicates the security level for non-secure non-invasive debug: 0b00 Functionality is not implemented or is controlled elsewhere.
[1:0]	NSID	Indicates the security level for non-secure invasive debug: 0b00 Functionality is not implemented or is controlled elsewhere.

3.7.13 Device Configuration Register

The DEVID Register characteristics are:

Purpose Indicates the capabilities of the component.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-35 on page 3-34.

Figure 3-45 shows the bit assignments.

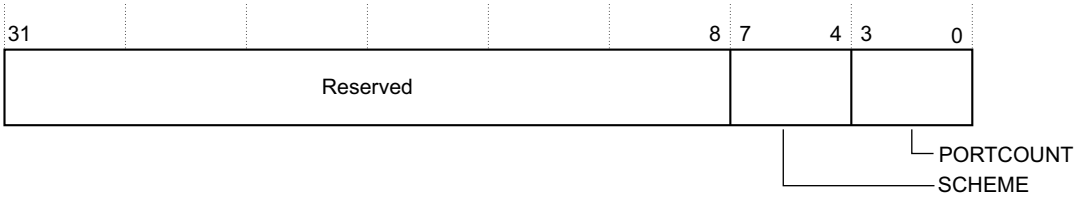


Figure 3-45 DEVID Register bit assignments

Table 3-48 shows the bit assignments.

Table 3-48 DEVID Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	SCHEME	Indicates the priority scheme implemented in this component. 0b0011 Program the slave ports to have higher or lower priority with respect to each other.
[3:0]	PORTCOUNT	Indicates the number of input ports connected. 0x0 and 0x1 are illegal values. 0b0010 Two ATB slave ports are present. 0b0011 Three ATB slave ports are present. 0b0100 Four ATB slave ports are present. 0b0101 Five ATB slave ports are present. 0b0110 Six ATB slave ports are present. 0b0111 Seven ATB slave ports are present. 0b1000 Eight ATB slave ports are present.

3.7.14 Device Type Identifier Register

The DEVTYPE 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 See the register summary in Table 3-35 on page 3-34.

Figure 3-46 shows the bit assignments.

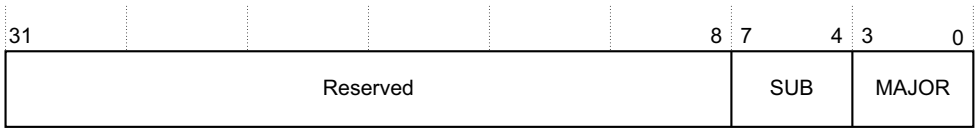


Figure 3-46 DEVTYPE Register bit assignments

Table 3-49 shows the bit assignments.

Table 3-49 DEVTYPE Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	SUB	Sub-classification of the type of the debug component as specified in the *CoreSight Architecture Specification* within the major classification as specified in the MAJOR field: `0b0001` This component arbitrates ATB inputs mapping to ATB outputs.
[3:0]	MAJOR	Major classification of the type of the debug component as specified in the *CoreSight Architecture Specification* for this debug and trace component: `0b0010` This component has both ATB inputs and ATB outputs.

3.7.15 Peripheral ID4 Register

The PIDR4 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-35 on page 3-34.

Figure 3-47 shows the bit assignments.

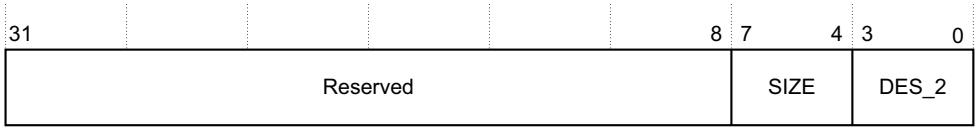


Figure 3-47 PIDR4 bit assignments

Table 3-50 shows the bit assignments.

Table 3-50 PIDR4 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	SIZE	This is a 4-bit value that indicates the total contiguous size of the memory window that this component uses in powers of 2 from the standard 4KB. `0b0000` Indicates that the device only occupies 4KB of memory.
[3:0]	DES_2	This is the JEDEC JEP106 continuation code. This code, along with bits[6:4] of the identity code defined in the PIDR0 Register and bits[3:0] of the identity code defined in the PIDR1 Register, identifies the designer of the component. `0b0100` JEDEC continuation code.

3.7.16 Peripheral ID5-7 registers

The PIDR5-7 characteristics are:

Purpose Reserved.

Usage constraints There are no usage constraints.

Configurations These registers are available in all configurations.

Attributes See the register summary in Table 3-35 on page 3-34.

Figure 3-48 shows the bit assignments.

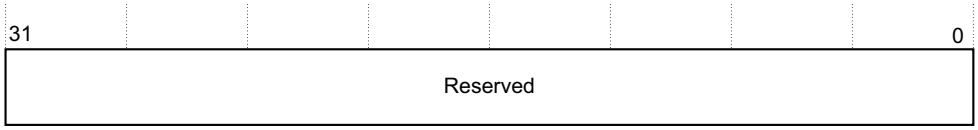


Figure 3-48 PIDR5-7 bit assignments

Table 3-51 shows the bit assignments.

Table 3-51 PIDR5-7 bit assignments

Bits	Name	Function
[31:0]	Reserved	-

3.7.17 Peripheral ID0 Register

The PIDR0 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 See the register summary in Table 3-35 on page 3-34.

Figure 3-49 on page 3-53 shows the bit assignments.

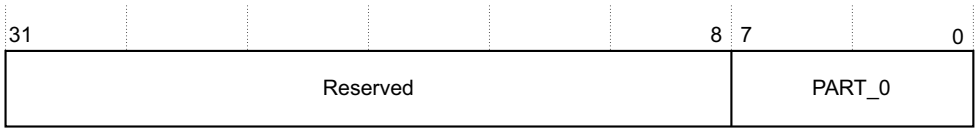


Figure 3-49 PIDR0 bit assignments

Table 3-52 shows the bit assignments.

Table 3-52 PIDR0 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PART_0	Bits[7:0] of the 12-bit part number of the component. The designer of the component assigns this part number. `0x08` Indicates bits[7:0] of the part number of the component.

3.7.18 Peripheral ID1 Register

The PIDR1 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 See the register summary in Table 3-35 on page 3-34.

Figure 3-50 shows the PIDR1 bit assignments.

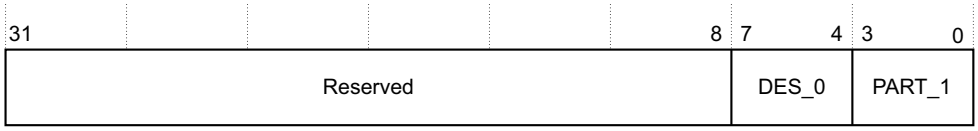


Figure 3-50 PIDR1 bit assignments

Table 3-53 shows the PIDR1 bit assignments.

Table 3-53 PIDR1 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	DES_0	Bits[3:0] of the JEDEC JEP106 identity code. This code, along with bits[6:4] of the identity code defined in the PIDR2 and the continuation code defined in the PIDR4, identifies the designer of the component. `0b1011` Bits[3:0] of the JEDEC JEP106 Identity Code. The default value is ARM.
[3:0]	PART_1	Bits[11:8] of the 12-bit part number of the component. The designer of the component assigns this part number. `0b1001` Indicates bits[11:8] of the part number of the component.

3.7.19 Peripheral ID2 Register

The PIDR2 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 See the register summary in Table 3-35 on page 3-34.

Figure 3-51 shows the PIDR2 bit assignments.

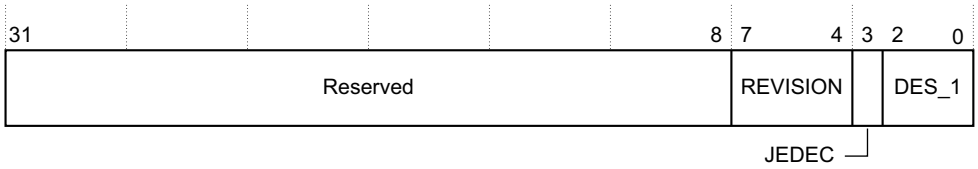


Figure 3-51 PIDR2 bit assignments

Table 3-54 shows the PIDR2 bit assignments.

Table 3-54 PIDR2 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	REVISION	An incremental value starting from `0b0000` for the first revision of this component. This increases by 1 for both major and minor revisions and is used to identify the major or minor revisions. `0b0010` See Table 1-1 on page 1-4 for the device version.
[3]	JEDEC	Always set. Indicates whether the JEDEC assigned designer ID is used. `0b1` Indicates that a JEDEC assigned designer ID is used.
[2:0]	DES_1	Bits[6:4] of the JEDEC JEP106 identity code. This code, along with bits[3:0] of the identity code defined in the PIDR1 and the continuation code defined in the PIDR4, identifies the designer of the component. `0b011` Bits[6:4] of the JEDEC JEP106 Identity Code. The default value is ARM.

3.7.20 Peripheral ID3 Register

The PIDR3 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-35 on page 3-34.

Figure 3-52 on page 3-55 shows the PIDR3 bit assignments.

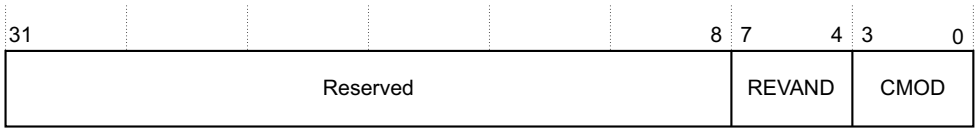


Figure 3-52 PIDR3 bit assignments

Table 3-55 shows the PIDR3 bit assignments.

Table 3-55 PIDR3 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	REVAND	Indicates minor errata fixes specific to the revision of the component being used, for example metal fixes after implementation. In most cases, this field is zero. ARM recommends that the component designers ensure that a metal fix can change this field if required, for example, by driving it from registers that reset to zero. `0b0000` Indicates that there are no errata fixes to this component.
[3:0]	CMOD	Indicates whether the customer has modified the behavior of the component. In most cases, this field is zero. The customer changes this value on modifications to this component. `0b0000` Indicates that the customer has not modified this component.

3.7.21 Component ID0 Register

The CIDR0 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-35 on page 3-34.

Figure 3-53 shows the bit assignments.

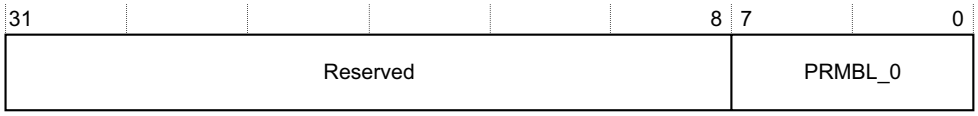


Figure 3-53 CIDR0 bit assignments

Table 3-56 shows the bit assignments.

Table 3-56 CIDR0 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_0	Contains bits[7:0] of the component identification code. `0x0D` Bits[7:0] of the identification code.

3.7.22 Component ID1 Register

The CIDR1 characteristics are:

Purpose A component identification register that indicates 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 See the register summary in Table 3-35 on page 3-34.

Figure 3-54 shows the bit assignments.

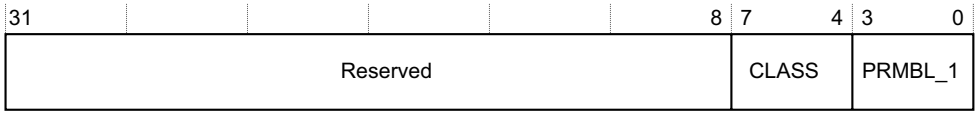


Figure 3-54 CIDR1 bit assignments

Table 3-57 shows the bit assignments.

Table 3-57 CIDR1 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	CLASS	Class of the component, for example, whether the component is a ROM table or a generic CoreSight component. Contains bits[15:12] of the component identification code. `0b1001` Indicates that the component is a CoreSight component.
[3:0]	PRMBL_1	Contains bits[11:8] of the component identification code. `0b0000` Bits[11:8] of the identification code.

3.7.23 Component ID2 Register

The CIDR2 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-35 on page 3-34.

Figure 3-55 shows the bit assignments.

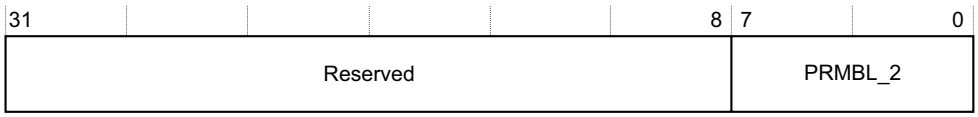


Figure 3-55 CIDR2 bit assignments

Table 3-58 shows the bit assignments.

Table 3-58 CIDR2 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_2	Contains bits[23:16] of the component identification code. `0x05` Bits[23:16] of the identification code.

3.7.24 Component ID3 Register

The CIDR3 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 See the register summary in Table 3-35 on page 3-34.

Figure 3-56 shows the bit assignments.

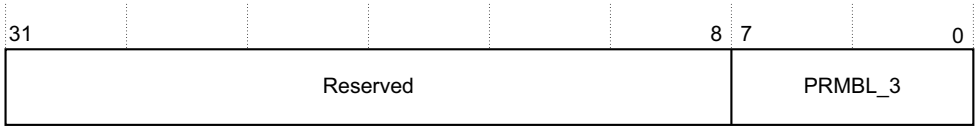


Figure 3-56 CIDR3 bit assignments

Table 3-59 shows the bit assignments.

Table 3-59 CIDR3 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_3	Contains bits[31:24] of the component identification code. `0xB1` Bits[31:24] of the identification code.

3.8 ATB replicator register summary

Table 3-60 shows the ATB replicator registers in offset order from the base memory address.

Table 3-60 ATB replicator register summary

Offset	Name	Type	Reset	Description
0x000	IDFILTER0	RW	0x00000000	*ID filtering for ATB master port 0* on page 3-59
0x004	IDFILTER1	RW	0x00000000	*ID filtering for ATB master port 1* on page 3-60
0xEFC	ITATBCTR0	WO	0x00000000	*Integration Mode ATB Control 0 Register* on page 3-61
0xEF8	ITATBCTR1	RO	0x00000000	*Integration Mode ATB Control 1 Register* on page 3-62
0xF00	ITCTRL	RW	0x00000000	*Integration Mode Control Register* on page 3-63
0xFA0	CLAIMSET	RW	0x0000000F	*Claim Tag Set Register* on page 3-64
0xFA4	CLAIMCLR	RW	0x00000000	*Claim Tag Clear Register* on page 3-65
0xFB0	LAR	WO	0x00000000	*Lock Access Register* on page 3-65
0xFB4	LSR	RO	0x00000003	*Lock Status* on page 3-66
0xFB8	AUTHSTATUS	RO	0x00000000	*Authentication Status Register* on page 3-67
0xFC8	DEVID	RO	0x00000002	*Device Configuration Register* on page 3-67
0xFCC	DEVTYPE	RO	0x00000022	*Device Type Identifier Register* on page 3-68
0xFD0	PIDR4	RO	0x00000004	*Peripheral ID4 Register* on page 3-69
0xFD4	PIDR5	RO	0x00000000	*Peripheral ID5-7 Registers* on page 3-69
0xFD8	PIDR6	RO	0x00000000	
0xFDC	PIDR7	RO	0x00000000	
0xFE0	PIDR0	RO	0x00000009	*Peripheral ID0 Register* on page 3-70
0xFE4	PIDR1	RO	0x000000B9	*Peripheral ID1 Register* on page 3-70
0xFE8	PIDR2	RO	0x0000001B	*Peripheral ID2 Register* on page 3-71
0xFEC	PIDR3	RO	0x00000000	*Peripheral ID3 Register* on page 3-72
0xFF0	CIDR0	RO	0x0000000D	*Component ID0 Register* on page 3-72
0xFF4	CIDR1	RO	0x00000090	*Component ID1 Register* on page 3-73
0xFF8	CIDR2	RO	0x00000005	*Component ID2 Register* on page 3-74
0xFFC	CIDR3	RO	0x000000B1	*Component ID3 Register* on page 3-74

3.9 ATB replicator register descriptions

This section describes the ATB replicator registers. Table 3-60 on page 3-58 provides cross-references to individual registers.

The registers are only present if you select the APB programming interface.

3.9.1 ID filtering for ATB master port 0

The IDFILTER0 Register characteristics are:

Purpose Enables the programming of ID filtering for master port 0.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-60 on page 3-58.

Figure 3-57 shows the bit assignments.

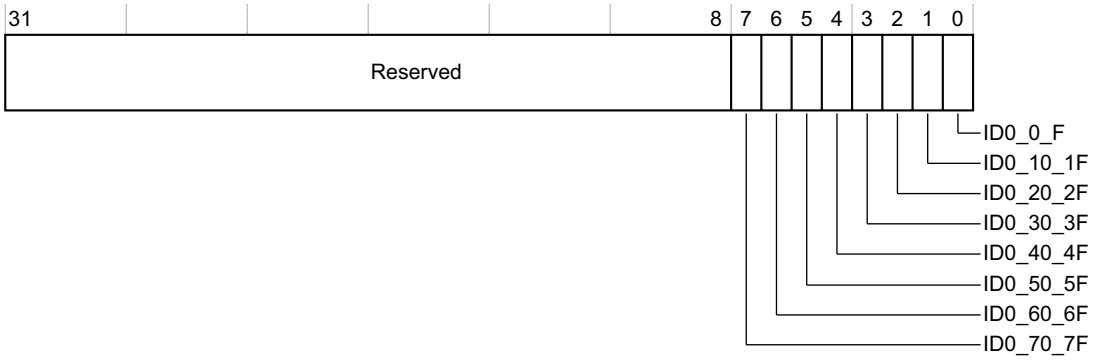


Figure 3-57 IDFILTER0 Register bit assignments

Table 3-61 shows the bit assignments.

Table 3-61 IDFILTER0 Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7]	ID0_70_7F	Enable or disable ID filtering for IDs `0x70-0x7F`. **0** Transactions with these IDs are passed on to ATB master port 0. **1** Transactions with these IDs are discarded by the replicator.
[6]	ID0_60_6F	Enable or disable ID filtering for IDs `0x60-0x6F`. **0** Transactions with these IDs are passed on to ATB master port 0. **1** Transactions with these IDs are discarded by the replicator.
[5]	ID0_50_5F	Enable or disable ID filtering for IDs `0x50-0x5F`. **0** Transactions with these IDs are passed on to ATB master port 0. **1** Transactions with these IDs are discarded by the replicator.

Table 3-61 IDFILTER0 Register bit assignments (continued)

Bits	Name	Function
[4]	ID0_40_4F	Enable or disable ID filtering for IDs `0x40-0x4F`. **0** Transactions with these IDs are passed on to ATB master port 0. **1** Transactions with these IDs are discarded by the replicator.
[3]	ID0_30_3F	Enable or disable ID filtering for IDs `0x30-0x3F`. **0** Transactions with these IDs are passed on to ATB master port 0. **1** Transactions with these IDs are discarded by the replicator.
[2]	ID0_20_2F	Enable or disable ID filtering for IDs `0x20-0x2F`. **0** Transactions with these IDs are passed on to ATB master port 0. **1** Transactions with these IDs are discarded by the replicator.
[1]	ID0_10_1F	Enable or disable ID filtering for IDs `0x10-0x1F`. **0** Transactions with these IDs are passed on to ATB master port 0. **1** Transactions with these IDs are discarded by the replicator.
[0]	ID0_0_F	Enable or disable ID filtering for IDs `0x0-0xF`. **0** Transactions with these IDs are passed on to ATB master port 0. **1** Transactions with these IDs are discarded by the replicator.

3.9.2 ID filtering for ATB master port 1

The IDFILTER1 Register characteristics are:

Purpose Enables the programming of ID filtering for master port 1.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-60 on page 3-58.

Figure 3-58 shows the bit assignments.

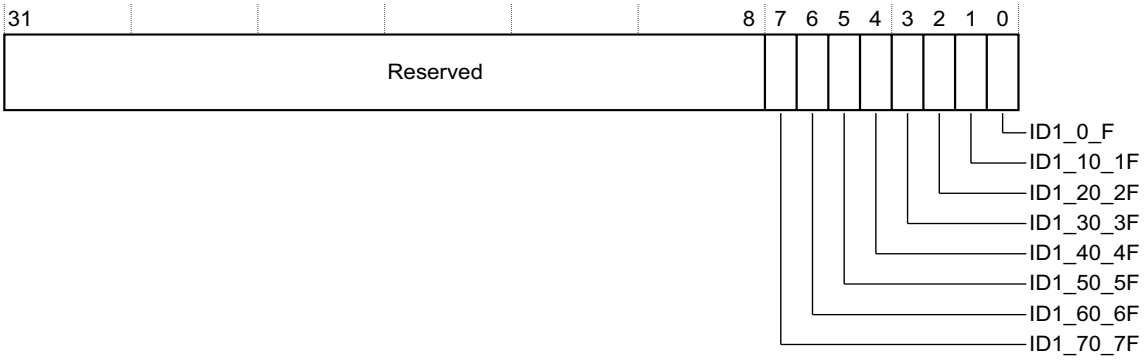


Figure 3-58 IDFILTER1 Register bit assignments

Table 3-62 shows the bit assignments.

Table 3-62 IDFILTER1 Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7]	ID1_70_7F	Enable or disable ID filtering for IDs `0x70-0x7F`. **0** Transactions with these IDs are passed on to ATB master port 1. **1** Transactions with these IDs are discarded by the replicator.
[6]	ID1_60_6F	Enable or disable ID filtering for IDs `0x60-0x6F`. **0** Transactions with these IDs are passed on to ATB master port 1. **1** Transactions with these IDs are discarded by the replicator.
[5]	ID1_50_5F	Enable or disable ID filtering for IDs `0x50-0x5F`. **0** Transactions with these IDs are passed on to ATB master port 1. **1** Transactions with these IDs are discarded by the replicator.
[4]	ID1_40_4F	Enable or disable ID filtering for IDs `0x40-0x4F`. **0** Transactions with these IDs are passed on to ATB master port 1. **1** Transactions with these IDs are discarded by the replicator.
[3]	ID1_30_3F	Enable or disable ID filtering for IDs `0x30-0x3F`. **0** Transactions with these IDs are passed on to ATB master port 1. **1** Transactions with these IDs are discarded by the replicator.
[2]	ID1_20_2F	Enable or disable ID filtering for IDs `0x20-0x2F`. **0** Transactions with these IDs are passed on to ATB master port 1. **1** Transactions with these IDs are discarded by the replicator.
[1]	ID1_10_1F	Enable or disable ID filtering for IDs `0x10-0x1F`. **0** Transactions with these IDs are passed on to ATB master port 1. **1** Transactions with these IDs are discarded by the replicator.
[0]	ID1_0_F	Enable or disable ID filtering for IDs `0x0-0xF`. **0** Transactions with these IDs are passed on to ATB master port 1. **1** Transactions with these IDs are discarded by the replicator.

3.9.3 Integration Mode ATB Control 0 Register

The ITATBCTR0 Register characteristics are:

Purpose Controls the value of the **ATVALIDM0**, **ATVALIDM1**, and **ATREADYS** outputs in integration mode.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-60 on page 3-58.

Figure 3-59 shows the bit assignments.

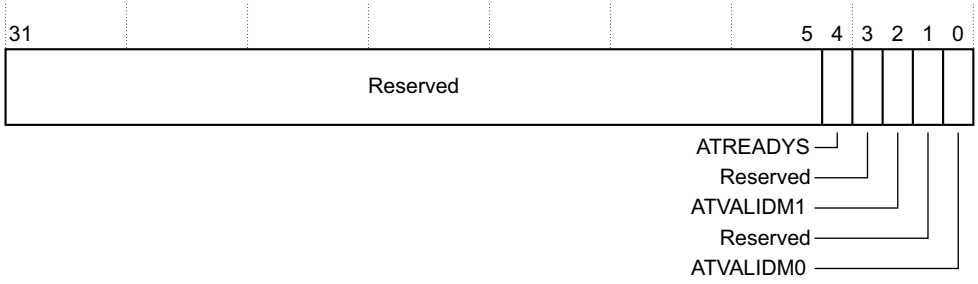


Figure 3-59 ITATBCTR0 Register bit assignments

Table 3-63 shows the bit assignments.

Table 3-63 ITATBCTR0 Register bit assignments

Bits	Name	Function
[31:5]	Reserved	-
[4]	ATREADYS	Sets the value of the **ATREADYS** output. **0** Drive logic 0 on the **ATREADYS** output. **1** Drive logic 1 on the **ATREADYS** output.
[3]	Reserved	-
[2]	ATVALIDM1	Sets the value of the **ATVALIDM1** output. **0** Drive logic 0 on the **ATVALIDM1** output. **1** Drive logic 1 on the **ATVALIDM1** output.
[1]	Reserved	-
[0]	ATVALIDM0	Sets the value of the **ATVALIDM0** output. **0** Drive logic 0 on the **ATVALIDM0** output. **1** Drive logic 1 on the **ATVALIDM0** output.

3.9.4 Integration Mode ATB Control 1 Register

The ITATBCTR1 Register characteristics are:

Purpose Returns the value of the **ATREADYM0**, **ATREADYM1**, and **ATVALIDS** inputs in integration mode.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-60 on page 3-58.

Figure 3-60 on page 3-63 shows the bit assignments.

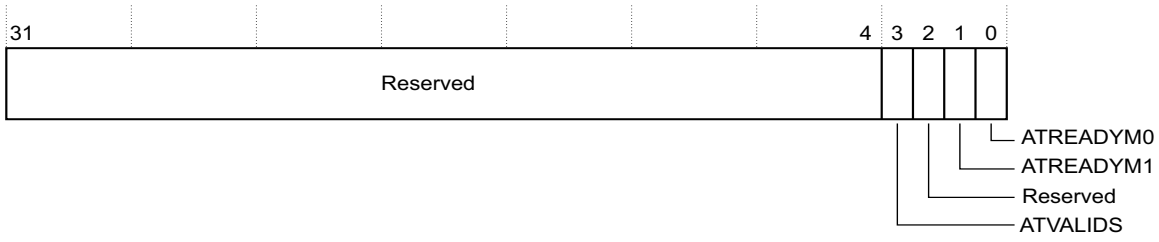


Figure 3-60 ITATBCTR1 Register bit assignments

Table 3-64 shows the bit assignments.

Table 3-64 ITATBCTR1 Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3]	ATVALIDS	Reads the value of the **ATVALIDS** input. `0b1` Pin is at logic 1. `0b0` Pin is at logic 0.
[2]	Reserved	-
[1]	ATREADYM1	Reads the value of the **ATREADYM1** input. `0b1` Pin is at logic 1. `0b0` Pin is at logic 0.
[0]	ATREADYM0	Reads the value of the **ATREADYM0** input. `0b1` Pin is at logic 1. `0b0` Pin is at logic 0.

3.9.5 Integration Mode Control Register

The ITCTRL Register characteristics are:

Purpose Enables topology detection. For more information, see the *CoreSight Architecture Specification*. This register enables the component to switch from a functional mode, which is the default behavior, to integration mode where the inputs and outputs of the component can be directly controlled for the purposes of integration testing and topology detection.

Note

When a device is in integration mode, the intended functionality might not be available.

After performing integration or topology detection, you must reset the system to ensure correct behavior of CoreSight and other connected system components that the integration or topology detection can affect.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-60 on page 3-58.

Figure 3-61 on page 3-64 shows the bit assignments.

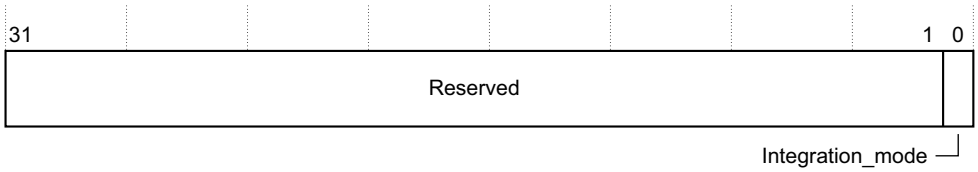


Figure 3-61 ITCTRL Register bit assignments

Table 3-65 shows the bit assignments.

Table 3-65 ITCTRL Register bit assignments

Bits	Name	Function
[31:1]	Reserved	-
[0]	IME	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. **0** Disable integration mode. **1** Enable integration mode.

3.9.6 Claim Tag Set Register

The CLAIMSET Register characteristics are:

Purpose Software can use the claim tag to coordinate application and debugger access to trace unit functionality. The claim tags have no effect on the operation of the component. The CLAIMSET Register sets bits in the claim tag, and determines the number of claim bits implemented.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-60 on page 3-58.

Figure 3-62 shows the bit assignments.

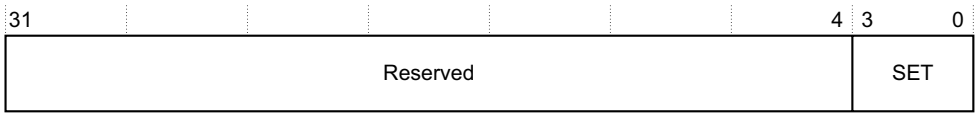


Figure 3-62 CLAIMSET Register bit assignments

Table 3-66 shows the bit assignments.

Table 3-66 CLAIMSET Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	SET_W	When you write 0 to bit[n] of this register, there is no effect on the claim tag bit[n]. When you write 1 to bit[n] of this register, bit[n] in the claim tag is set to 1.
[3:0]	SET_R	When you read 0 on a particular bit of this register, it indicates that this claim tag bit is not implemented. When you read 1 on a particular bit of this register, it indicates that this claim tag bit is implemented. `0b1111` Indicates that four bits of claim tag are implemented.

3.9.7 Claim Tag Clear Register

The CLAIMCLR Register characteristics are:

Purpose Software can use the claim tag to coordinate application and debugger access to trace unit functionality. The claim tags have no effect on the operation of the component. The CLAIMCLR Register clears the bits in the claim tag, and determines the current value of the claim tag.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-60 on page 3-58.

Figure 3-63 shows the bit assignments.

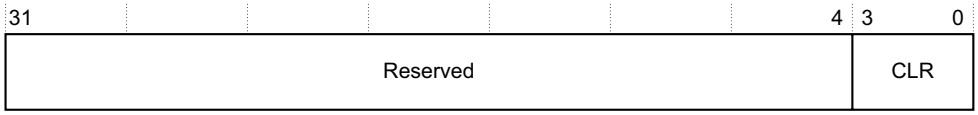


Figure 3-63 CLAIMCLR Register bit assignments

Table 3-67 shows the bit assignments.

Table 3-67 CLAIMCLR Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	CLR_W	When you write 0 to bit[n] of this register, there is no effect on the claim tag bit[n]. When you write 1 to bit[n] of this register, bit[n] in the claim tag is cleared to 0.
[3:0]	CLR_R	The value present reflects the present value of the Claim Tag.

3.9.8 Lock Access Register

The LAR characteristics are:

Purpose Controls write access from self-hosted, on-chip accesses. The LAR does not affect the accesses using the external debugger interface.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-60 on page 3-58.

Figure 3-64 shows the bit assignments.

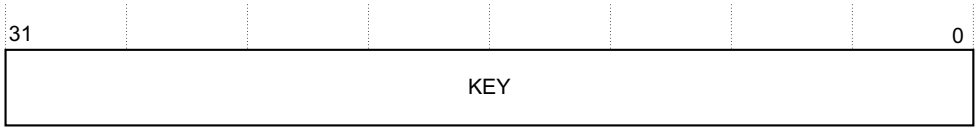


Figure 3-64 LAR bit assignments

Table 3-68 shows the bit assignments.

Table 3-68 LAR bit assignments

Bits	Name	Function
[31:0]	KEY	When you write `0xC5ACCE55`, subsequent write operations to this device are enabled. Any other value disables subsequent write operations.

3.9.9 Lock Status

The LSR characteristics are:

Purpose Indicates the status of the lock control mechanism. This lock prevents accidental writes by code being debugged. When locked, write accesses are denied for all registers except for the LAR. The lock registers do not affect accesses from the external debug interface. This register reads as 0 when accessed from the external debug interface.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-60 on page 3-58.

Figure 3-65 shows the bit assignments.

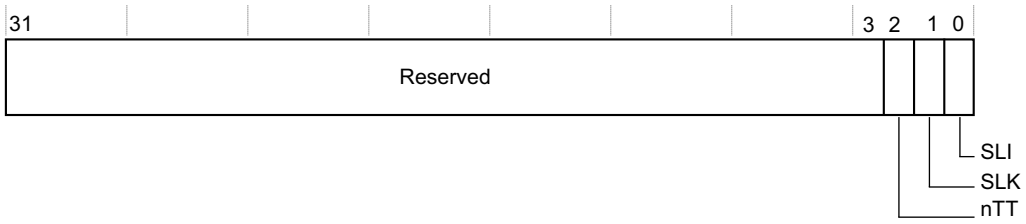


Figure 3-65 LSR bit assignments

Table 3-69 shows the bit assignments.

Table 3-69 LSR bit assignments

Bits	Name	Function
[31:3]	Reserved	-
[2]	nTT	Indicates that the LAR is implemented as 32-bit. `0b0` This component implements a 32-bit LAR Register.
[1]	SLK	Returns the present lock status of the device. `0b0` Indicates that write operations are permitted in this device. `0b1` Indicates that write operations are not permitted in this device. Read operations are permitted.
[0]	SLI	Indicates that a lock control mechanism is present in this device. `0b0` Indicates that a lock control mechanism is not present in this device. Write operations to the LAR are ignored. `0b1` Indicates that a lock control mechanism is present in this device.

3.9.10 Authentication Status Register

The AUTHSTATUS Register characteristics are:

Purpose Reports the required security level and present status.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-60 on page 3-58.

Figure 3-66 shows the bit assignments.

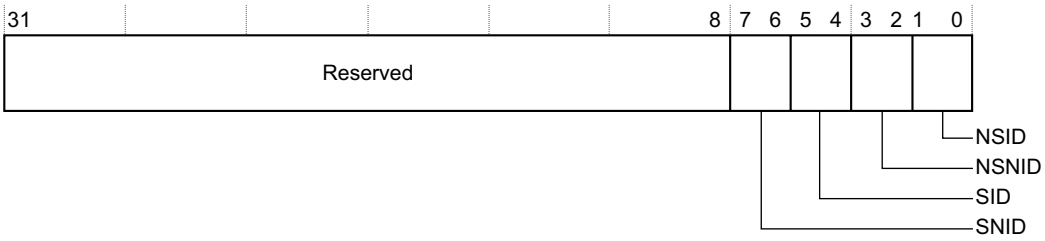


Figure 3-66 AUTHSTATUS Register bit assignments

Table 3-70 shows the bit assignments.

Table 3-70 AUTHSTATUS Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:6]	SNID	Indicates the security level for secure non-invasive debug: 0b00 Functionality is not implemented or is controlled elsewhere.
[5:4]	SID	Indicates the security level for secure invasive debug: 0b00 Functionality is not implemented or is controlled elsewhere.
[3:2]	NSNID	Indicates the security level for non-secure non-invasive debug: 0b00 Functionality is not implemented or is controlled elsewhere.
[1:0]	NSID	Indicates the security level for non-secure invasive debug: 0b00 Functionality is not implemented or is controlled elsewhere.

3.9.11 Device Configuration Register

The DEVID Register characteristics are:

Purpose Indicates the capabilities of the component.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-60 on page 3-58.

Figure 3-67 on page 3-68 shows the bit assignments.

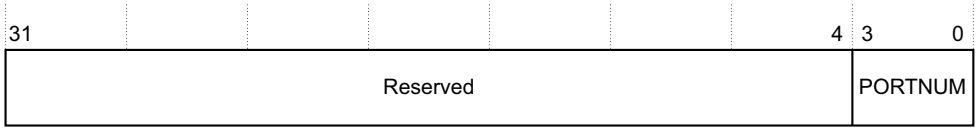


Figure 3-67 DEVID Register bit assignments

Table 3-71 shows the bit assignments.

Table 3-71 DEVID Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	PORTNUM	Indicates the number of master ports implemented. `0b0010` Two master ports are implemented.

3.9.12 Device Type Identifier Register

The DEVTYPE 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 See the register summary in Table 3-60 on page 3-58.

Figure 3-68 shows the bit assignments.

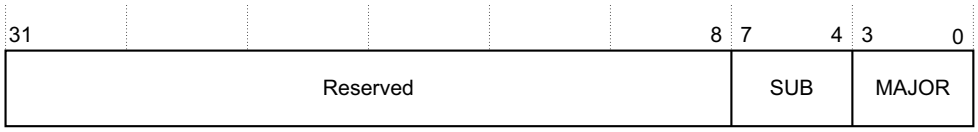


Figure 3-68 DEVTYPE Register bit assignments

Table 3-72 shows the bit assignments.

Table 3-72 DEVTYPE Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	SUB	Sub-classification of the type of the debug component as specified in the *CoreSight Architecture Specification* within the major classification as specified in the MAJOR field. `0b0010` Indicates that this component replicates traces from single source to multiple targets.
[3:0]	MAJOR	Major classification of the type of the debug component as specified in the *CoreSight Architecture Specification* for this debug and trace component. `0b0010` Indicates that this component has ATB input and output.

3.9.13 Peripheral ID4 Register

The PIDR4 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-60 on page 3-58.

Figure 3-69 shows the bit assignments.

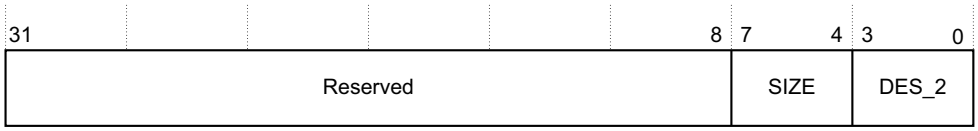


Figure 3-69 PIDR4 bit assignments

Table 3-73 shows the bit assignments.

Table 3-73 PIDR4 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	SIZE	This is a 4-bit value that indicates the total contiguous size of the memory window this component uses in powers of 2 from the standard 4KB. `0b0000` Indicates that the device only occupies 4KB of memory.
[3:0]	DES_2	This is the JEDEC JEP106 continuation code. This code, along with bits[6:4] of the identity code defined in the PIDR0 and bits[3:0] of the identity code defined in the PIDR1, identifies the designer of the component. `0b0100` JEDEC continuation code.

3.9.14 Peripheral ID5-7 Registers

The PIDR5-7 characteristics are:

Purpose Reserved.

Usage constraints There are no usage constraints.

Configurations These registers are available in all configurations.

Attributes See the register summary in Table 3-60 on page 3-58.

Figure 3-70 shows the bit assignments.

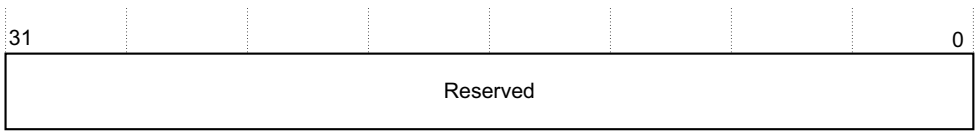


Figure 3-70 PIDR5-7 bit assignments

Table 3-74 shows the bit assignments.

Table 3-74 PIDR5-7 bit assignments

Bits	Name	Function
[31:0]	Reserved	-

3.9.15 Peripheral ID0 Register

The PIDR0 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 See the register summary in Table 3-60 on page 3-58.

Figure 3-71 shows the bit assignments.

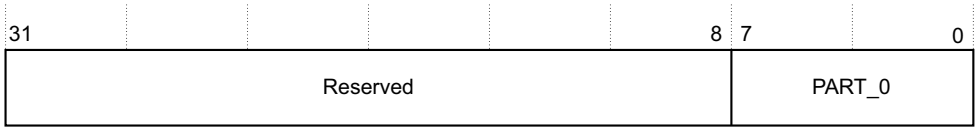


Figure 3-71 PIDR0 bit assignments

Table 3-75 shows the bit assignments.

Table 3-75 PIDR0 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PART_0	Bits[7:0] of the 12-bit part number of the component. The designer of the component assigns this part number. `0x09` Indicates bits[7:0] of the part number of the component.

3.9.16 Peripheral ID1 Register

The PIDR1 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 See the register summary in Table 3-60 on page 3-58.

Figure 3-72 on page 3-71 shows the bit assignments.

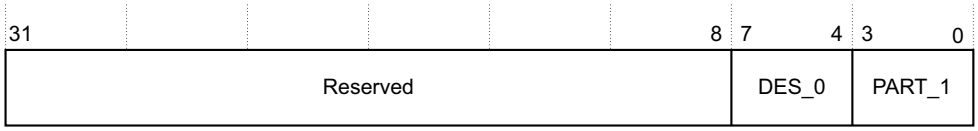


Figure 3-72 PIDR1 bit assignments

Table 3-76 shows the bit assignments.

Table 3-76 PIDR1 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	DES_0	Bits[3:0] of the JEDEC JEP106 identity code. This code, along with bits[6:4] of the identity code defined in the PIDR2 and the continuation code defined in the PIDR4, identifies the designer of the component. `0b1011` Bits[3:0] of the JEDEC JEP106 Identity Code. The default value is ARM.
[3:0]	PART_1	Bits[11:8] of the 12-bit part number of the component. The designer of the component assigns this part number. `0b1001` Indicates bits[11:8] of the part number of the component.

3.9.17 Peripheral ID2 Register

The PIDR2 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 See the register summary in Table 3-60 on page 3-58.

Figure 3-73 shows the bit assignments.

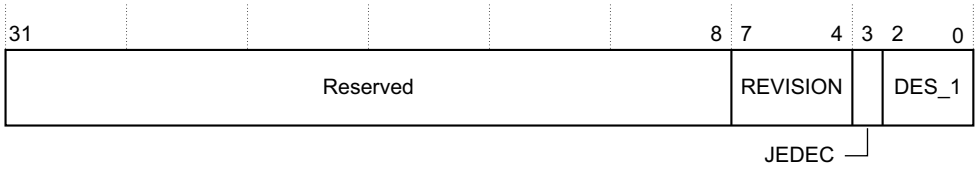


Figure 3-73 PIDR2 bit assignments

Table 3-77 shows the bit assignments.

Table 3-77 PIDR2 bit assignments

Bits	Name	Function
[31:8]	Reserved	-

Table 3-77 PIDR2 bit assignments (continued)

Bits	Name	Function
[7:4]	REVISION	An incremental value starting from 0b0000 for the first revision of this component. This increases by 1 for both major and minor revisions and is used to identify the major or minor revisions. 0b0001 This device is at r1p0.
[3]	JEDEC	Always set. Indicates whether the JEDEC assigned designer ID is used. 0b1 Indicates that a JEDEC assigned designer ID is used.
[2:0]	DES_1	Bits[6:4] of the JEDEC JEP106 identity code. This code, along with bits[3:0] of the identity code defined in the PIDR1 and the continuation code defined in the PIDR4, identifies the designer of the component. 0b011 Bits[6:4] of the JEDEC JEP106 Identity Code. The default value is ARM.

3.9.18 Peripheral ID3 Register

The PIDR3 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-60 on page 3-58.

Figure 3-74 shows the bit assignments.

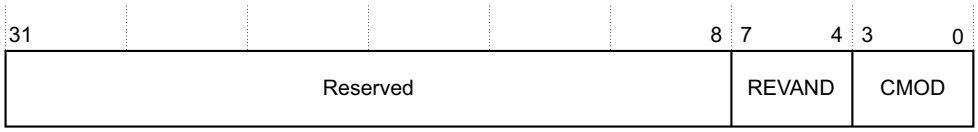


Figure 3-74 PIDR3 bit assignments

Table 3-78 shows the bit assignments.

Table 3-78 PIDR3 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	REVAND	Indicates minor errata fixes specific to the revision of the component being used, for example metal fixes after implementation. In most cases, this field is zero. ARM recommends that the component designers ensure that a metal fix can change this field if required, for example, by driving it from registers that reset to zero. 0b0000 Indicates that there are no errata fixes to this component.
[3:0]	CMOD	Indicates whether the customer has modified the behavior of the component. In most cases, this field is zero. The customer changes this value on modifications to this component. 0b0000 Indicates that the customer has not modified this component.

3.9.19 Component ID0 Register

The CIDR0 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-60 on page 3-58.

Figure 3-75 shows the bit assignments.

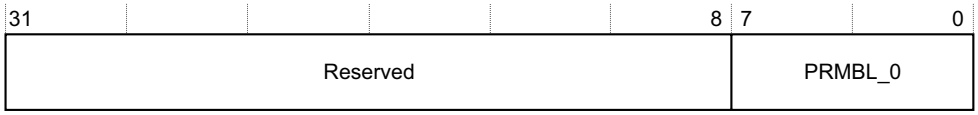


Figure 3-75 CIDR0 bit assignments

Table 3-79 shows the bit assignments.

Table 3-79 CIDR0 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_0	Contains bits[7:0] of the component identification code. `0x0D` Bits[7:0] of the identification code.

3.9.20 Component ID1 Register

The CIDR1 characteristics are:

Purpose A component identification Register that indicates 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 See the register summary in Table 3-60 on page 3-58.

Figure 3-76 shows the bit assignments.

Figure 3-76 CIDR1 bit assignments

Table 3-80 shows the bit assignments.

Table 3-80 CIDR1 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	CLASS	Class of the component, for example, whether the component is a ROM table or a generic CoreSight component. Contains bits[15:12] of the component identification code. `0b1001` Indicates that the component is a CoreSight component.
[3:0]	PRMBL_1	Contains bits[11:8] of the component identification code. `0b0000` Bits[11:8] of the identification code.

3.9.21 Component ID2 Register

The CIDR2 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-60 on page 3-58.

Figure 3-77 shows the bit assignments.

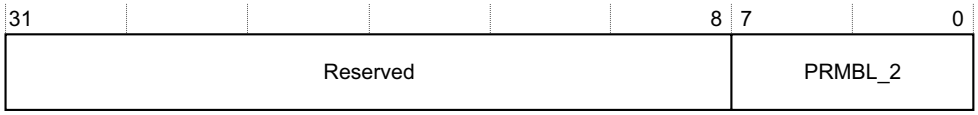


Figure 3-77 CIDR2 bit assignments

Table 3-81 shows the bit assignments.

Table 3-81 CIDR2 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_2	Contains bits[23:16] of the component identification code. `0x05` Bits[23:16] of the identification code.

3.9.22 Component ID3 Register

The CIDR3 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-60 on page 3-58.

Figure 3-78 shows the bit assignments.

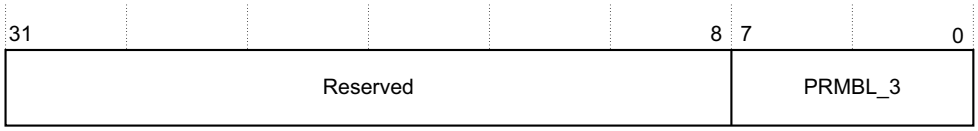


Figure 3-78 CIDR3 bit assignments

Table 3-82 shows the bit assignments.

Table 3-82 CIDR3 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_3	Contains bits[31:24] of the component identification code. `0xB1` Bits[31:24] of the identification code.

3.10 ETB register summary

Table 3-83 shows the ETB registers in offset order from the base memory address.

Table 3-83 ETB register summary

Offset	Name	Type	Reset	Description
0x004	RDP	RO	0x00000000	*ETB RAM Depth Register* on page 3-78
0x00C	STS	RO	0x00000008	*ETB Status Register* on page 3-78
0x010	RRD	RO	0x00000000	*ETB RAM Read Data Register* on page 3-79
0x014	RRP	RW	0x00000000	*ETB RAM Read Pointer Register* on page 3-80
0x018	RWP	RW	0x00000000	*ETB RAM Write Pointer Register* on page 3-80
0x01C	TRG	RW	0x00000000	*ETB Trigger Counter Register* on page 3-81
0x020	CTL	RW	0x00000000	*ETB Control Register* on page 3-82
0x024	RWD	WO	0x00000000	*ETB RAM Write Data Register* on page 3-83
0x300	FFSR	RO	0x00000002	*ETB Formatter and Flush Status Register* on page 3-83
0x304	FFCR	RW	0x00000000	*ETB Formatter and Flush Control Register* on page 3-84
0xEE0	ITMISCOP0	WO	0x00000000	*Integration Test Miscellaneous Output Register 0* on page 3-86
0xEE4	ITTRFLINACK	WO	0x00000000	*Integration Test Trigger In and Flush In Acknowledge Register* on page 3-87
0xEE8	ITTRFLIN	RO	0x00000000	*Integration Test Trigger In and Flush In Register* on page 3-87
0xEEC	ITATBDATA0	RO	0x00000000	*Integration Test ATB Data Register 0* on page 3-88
0xEF0	ITATBCTR2	WO	0x00000000	*Integration Test ATB Control Register 2* on page 3-89
0xEF4	ITATBCTR1	RO	0x00000000	*Integration Test ATB Control Register 1* on page 3-90
0xEF8	ITATBCTR0	RO	0x00000000	*Integration Test ATB Control Register 0* on page 3-90
0xF00	ITCTRL	RW	0x00000000	*Integration Mode Control Register* on page 3-91
0xFA0	CLAIMSET	RW	0x0000000F	*Claim Tag Set Register* on page 3-92
0xFA4	CLAIMCLR	RW	0x00000000	*Claim Tag Clear Register* on page 3-92
0xFB0	LAR	WO	0x00000000	*Lock Access Register* on page 3-93
0xFB4	LSR	RO	0x00000003	*Lock Status Register* on page 3-94
0xFB8	AUTHSTATUS	RO	0x00000000	*Authentication Status Register* on page 3-94
0xFC8	DEVID	RO	0x00000000	*Device Configuration Register* on page 3-95
0xFCC	DEVTYPE	RO	0x00000021	*Device Type Identifier Register* on page 3-96
0xFD0	PIDR4	RO	0x00000004	*Peripheral ID4 Register* on page 3-97
0xFD4	PIDR5	RO	0x00000000	*Peripheral ID5-7 registers* on page 3-97
0xFD8	PIDR6	RO	0x00000000	
0xFDC	PIDR7	RO	0x00000000	
0xFE0	PIDR0	RO	0x00000007	*Peripheral ID0 Register* on page 3-98

Table 3-83 ETB register summary (continued)

Offset	Name	Type	Reset	Description
0xFE4	PIDR1	RO	0x000000B9	*Peripheral ID1 Register* on page 3-98
0xFE8	PIDR2	RO	0x0000003B	*Peripheral ID2 Register* on page 3-99
0xFEC	PIDR3	RO	0x00000000	*Peripheral ID3 Register* on page 3-100
0xFF0	CIDR0	RO	0x0000000D	*Component ID0 Register* on page 3-100
0xFF4	CIDR1	RO	0x00000090	*Component ID1 Register* on page 3-101
0xFF8	CIDR2	RO	0x00000005	*Component ID2 Register* on page 3-102
0xFFC	CIDR3	RO	0x000000B1	*Component ID3 Register* on page 3-102

3.11 ETB register descriptions

This section describes the ETB registers. Table 3-83 on page 3-76 provides cross references to individual registers.

3.11.1 ETB RAM Depth Register

The RDP Register characteristics are:

Purpose Defines the depth, in words, of the trace RAM.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-79 shows the bit assignments.

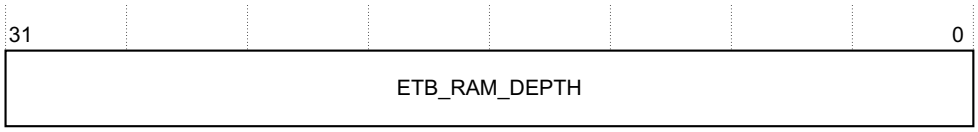


Figure 3-79 RDP Register bit assignments

Table 3-84 shows the bit assignments.

Table 3-84 RDP Register bit assignments

Bits	Name	Function
[31:0]	ETB_RAM_DEPTH	Defines the depth, in words, of the trace RAM.

3.11.2 ETB Status Register

The STS Register characteristics are:

Purpose Indicates the status of the ETB.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-80 shows the bit assignments.

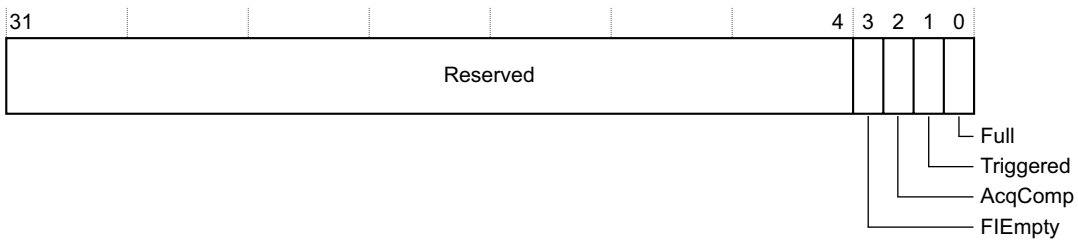


Figure 3-80 STS Register bit assignments

Table 3-85 shows the bit assignments.

Table 3-85 STS Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3]	FtEmpty	Formatter pipeline is empty. All data is stored to RAM. **0** Formatter pipeline is not empty. **1** Formatter pipeline is empty.
[2]	AcqComp	The acquisition complete flag indicates that the capture is completed when the formatter stops because of any of the methods defined in the Formatter and Flush Control Register, or TraceCaptEn = 0. This also results in the FtStopped bit in the Formatter and Flush Status Register going HIGH. **0** Acquisition is not complete. **1** Acquisition is complete.
[1]	Triggered	The Triggered bit is set when the component observes a trigger. This does not indicate that the formatter embedded a trigger in the trace data, but is determined when programming the Formatter and Flush Control Registers. **0** A trigger is not observed. **1** A trigger is observed.
[0]	Full	The flag indicates whether the RAM is full or not. **0** RAM write pointer is not wrapped around. The RAM is not full. **1** RAM write pointer is wrapped around. The RAM is full.

3.11.3 ETB RAM Read Data Register

The RRD Register characteristics are:

Purpose When trace capture is disabled, the contents of the ETB Trace RAM at the location addressed by the RAM Read Pointer Registers are placed in this register. Reading this register increments the RAM Read Pointer Register and triggers a RAM access cycle. If trace capture is enabled, FtStopped=0 and TraceCaptEn=1, and ETB RAM attempts a read operation, a read from this register outputs `0xFFFFFFFF` and the RAM Read Pointer Register does not auto-increment. A constant output of 1s corresponds to a synchronization output in the formatter protocol that is not applicable to the ETB, and so can be used to indicate a read error, when formatting is enabled.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-81 shows the bit assignments.

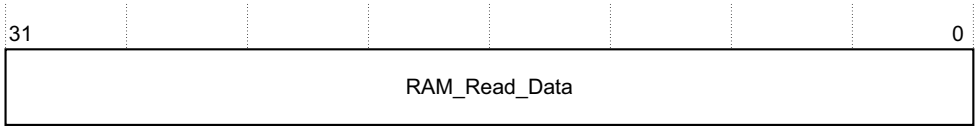


Figure 3-81 RRD Register bit assignments

Table 3-86 shows the bit assignments.

Table 3-86 RRD Register bit assignments

Bits	Name	Function
[31:0]	RAM_Read_Data	Data read from the ETB Trace RAM.

3.11.4 ETB RAM Read Pointer Register

The RRP Register characteristics are:

Purpose The RAM Read Pointer Register sets the read pointer used to the required value. When this register is written to, a RAM access is initiated. The RAM Read Data Register is then updated. You can also read this register to determine the present memory location that is referenced. You must not write to this register when trace capture is enabled, FtStopped=0 and TraceCaptEn=1. When trace capture is enabled, it is not possible to update the register even if there is a write operation.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-82 shows the bit assignments.

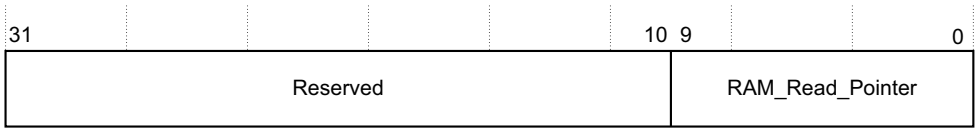


Figure 3-82 RRP Register bit assignments

Table 3-87 shows the bit assignments.

Table 3-87 RRP Register bit assignments

Bits	Name	Function
[31:10]	Reserved	-
[9:0]	RAM_Read_Pointer	Sets the read pointer to the required value. The read pointer is used to read entries from the Trace RAM through the APB interface.

3.11.5 ETB RAM Write Pointer Register

The RWP Register characteristics are:

Purpose The RAM Write Pointer Register sets the write pointer to the required value. The Write Pointer is used to write entries from the CoreSight bus to the Trace RAM. During trace capture, the pointer increments when the formatter asserts the DataValid flag. When this register wraps around from its maximum value to zero, the Full flag is set. You can also write to this register through APB to set the pointer for write accesses. You must not write to this register when trace capture is enabled, FtStopped=0 and TraceCaptEn=1. When trace capture is enabled it is not possible to update the register even if you do a write operation. You can also read this register

to determine the present memory location that is referenced. ARM recommends that addresses are 128-bit aligned when you use the formatter in normal or continuous modes.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-83 shows the bit assignments.

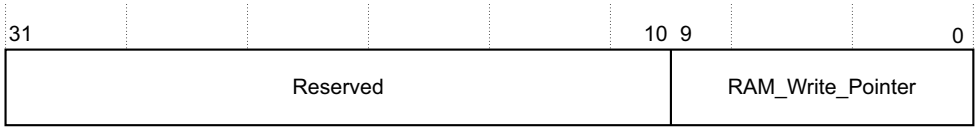


Figure 3-83 RWP Register bit assignments

Table 3-88 shows the bit assignments.

Table 3-88 RWP Register bit assignments

Bits	Name	Function
[31:10]	Reserved	-
[9:0]	RAM_Write_Pointer	The RAM Write Pointer Register sets the write pointer to the required value. The Write Pointer is used to write entries from the CoreSight bus to the Trace RAM.

3.11.6 ETB Trigger Counter Register

The TRG Register characteristics are:

Purpose Stops the formatter after a defined number of words are stored following the trigger event and disables write access to the trace RAM. The number of 32-bit words written to the trace RAM following the trigger event is equal to the value stored in this register plus 1.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-84 shows the bit assignments.

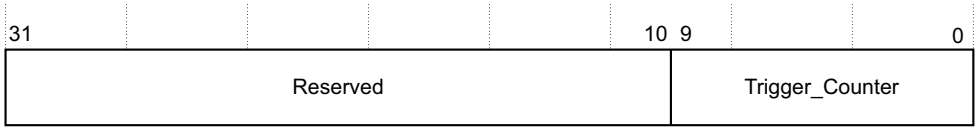


Figure 3-84 TRG Register bit assignments

Table 3-89 shows the bit assignments.

Table 3-89 TRG Register bit assignments

Bits	Name	Function
[31:10]	Reserved	-
[9:0]	Trigger_Counter	The counter is used as follows: **Trace after** The counter is set to a large value, slightly less than the number of entries in the RAM. **Trace before** The counter is set to a small value. **Trace about** The counter is set to half the depth of the trace RAM. You must not write to this register when trace capture is enabled, FtStopped=0 and TraceCaptEn=1. If a write is attempted, then the register is not updated. A read operation is permitted when trace capture is enabled.

3.11.7 ETB Control Register

The CTL Register characteristics are:

Purpose Controls trace capture by the ETB.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-85 shows the bit assignments.

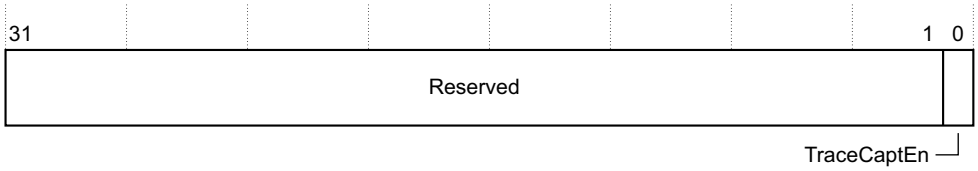


Figure 3-85 CTL Register bit assignments

Table 3-90 shows the bit assignments.

Table 3-90 CTL Register bit assignments

Bits	Name	Function
[31:1]	Reserved	-
[0]	TraceCaptEn	ETB Trace Capture Enable.This is the master enable bit forcing **FtStopped** HIGH when **TraceCaptEn** is LOW. When capture is disabled, any remaining data in the ATB formatter is stored to RAM. When all of the data are stored, the formatter outputs **FtStopped**. Capture is fully disabled, or complete, when **FtStopped** goes HIGH. See *ETB Formatter and Flush Status Register* on page 3-83. **0** Disable trace capture. **1** Enable trace capture.

3.11.8 ETB RAM Write Data Register

The RWD Register characteristics are:

Purpose Provides data that writes to ETB Trace RAM.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-86 shows the bit assignments.

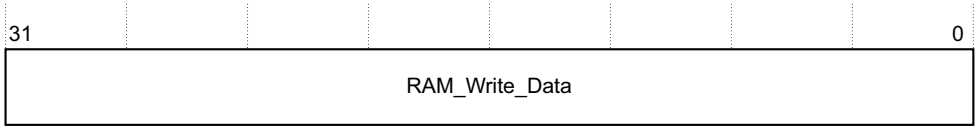


Figure 3-86 RWD Register bit assignments

Table 3-91 shows the bit assignments.

Table 3-91 RWD Register bit assignments

Bits	Name	Function
[31:0]	RAM_Write_Data	When the trace capture is disabled and writes the data to the ETB trace RAM, the contents of this register are placed into the ETB Trace RAM. Writing to this register increments the RAM Write Pointer Register value. If trace capture is enabled, and this register is accessed, then a read from this register outputs `0xFFFFFFFF`. Reads of this register never increment the RAM Write Pointer Register. A constant stream of 1s being output corresponds to a synchronization output from the ETB. If a write access is attempted, the data is not written into trace RAM.

3.11.9 ETB Formatter and Flush Status Register

The FFSR Register characteristics are:

Purpose Indicates the implemented trigger counter multipliers and other supported features of the trigger system.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-87 shows the bit assignments.

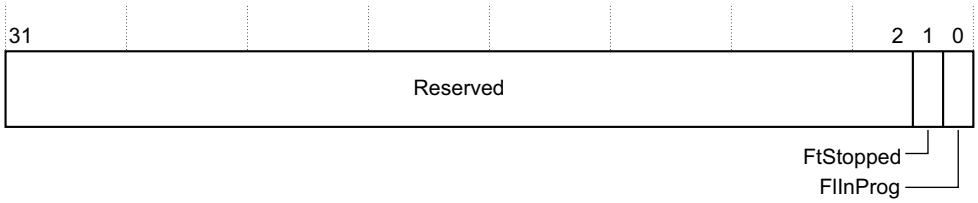


Figure 3-87 FFSR Register bit assignments

Table 3-92 shows the bit assignments.

Table 3-92 FFSR Register bit assignments

Bits	Name	Function
[31:2]	Reserved	-
[1]	FtStopped	Formatter stopped. The formatter has received a stop request signal and all trace data and post-amble is sent. Any additional trace data on the ATB interface is ignored and **atreadys** goes HIGH. **0** Formatter is not stopped. **1** Formatter is stopped.
[0]	FlInProg	Flush In Progress. This is an indication of the current state of **afvalids**. **0** **afvalids** is LOW. **1** **afvalids** is HIGH.

3.11.10 ETB Formatter and Flush Control Register

The FFCR Register characteristics are:

Purpose Controls the generation of stop, trigger, and flush events. To disable formatting and put the formatter into bypass mode, clear the bits 1 and 0. If both bits are set, then the formatter inserts triggers into the formatted stream. All three flush generating conditions can be enabled together. However, if a second or third flush event is generated then the current flush completes before the next flush is serviced. Flush from **flushin** takes priority over flush from trigger, which in turn completes before a manually activated flush. All trigger indication conditions can be enabled simultaneously although this can cause the appearance of multiple triggers, if flush using trigger is also enabled. Both `Stop On` settings can be enabled, although if flush on trigger, FOnTrig, is set up then none of the flushed data is stored. When the system stops, it returns **atready** and does not store the accepted data packets. This is to avoid stalling of any other connected devices using a trace replicator. If an event in the Formatter and Flush Control Register is required, it must be enabled before the originating event starts. Because requests from flushes and triggers can originate in an asynchronous clock domain, the exact time the component acts on the request cannot be determined with respect to configuring the control.

Note

To perform a stop on flush completion through a manually generated flush request, two write operations to the register are required:

- To enable the stop event, if it is not already enabled.
- To generate the manual flush.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-88 on page 3-85 shows the bit assignments.

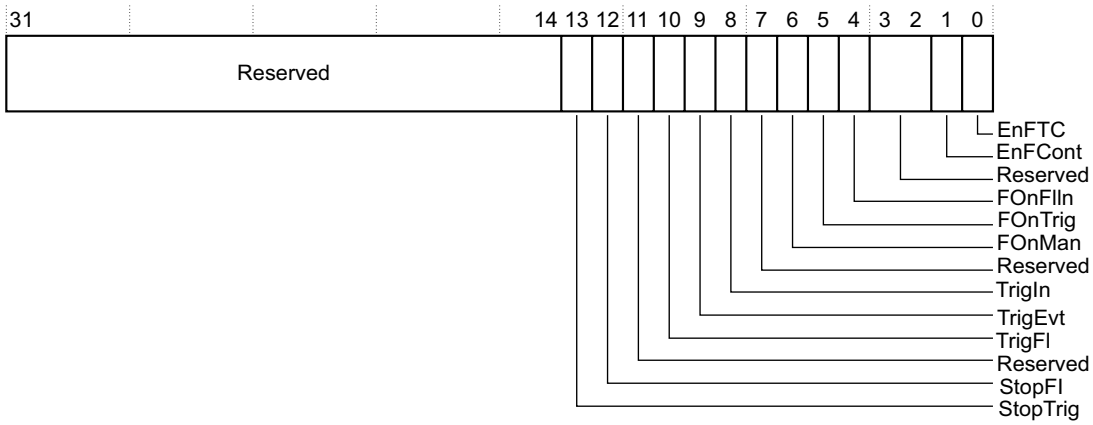


Figure 3-88 FFCR Register bit assignments

Table 3-93 shows the bit assignments.

Table 3-93 FFCR Register bit assignments

Bits	Name	Function
[31:14]	Reserved	-
[13]	StopTrig	Stop the formatter after a trigger event is observed. Reset to disabled, that is, zero. **0** Disable stopping the formatter after a trigger event is observed. **1** Enable stopping the formatter after a trigger event is observed.
[12]	StopFl	This forces the FIFO to drain off any part-completed packets. Setting this bit enables this function but this is cleared on reset, that is, disabled. **0** Disable stopping the formatter on return of **afreadys**. **1** Enable stopping the formatter on return of **afreadys**.
[11]	Reserved	-
[10]	TrigFl	Indicates a trigger on flush completion, **afreadys** is returned. **0** Disable trigger indication when **afreadys** is returned. **1** Enable trigger indication when **afreadys** is returned.
[9]	TrigEvt	Indicate a trigger on a trigger event. **0** Disable trigger indication on a trigger event. **1** Enable trigger indication on a trigger event.
[8]	TrigIn	Indicate a trigger when **trigin** is asserted. **0** Disable trigger indication when **trigin** is asserted. **1** Enable trigger indication when **trigin** is asserted.
[7]	Reserved	-
[6]	FOnMan	Setting this bit initiates a manual flush. This is cleared after this flush is serviced. This bit is cleared on reset. **0** Manual flush is not initiated. **1** Manual flush is initiated.
[5]	FOnTrig	Sets this bit to flush the data in the system when a trigger event occurs. This bit is cleared on reset. **0** Disable flush generation when a trigger event occurs. **1** Enable flush generation when a trigger event occurs.

Table 3-93 FFCR Register bit assignments (continued)

Bits	Name	Function
[4]	FOnFlIn	Enables use of the **flushin** connection. This bit is cleared on reset. **0** Disable flush generation using the **flushin** interface. **1** Enable flush generation using the **flushin** interface.
[3:2]	Reserved	-
[1]	EnFCont	Continuous mode in the ETB corresponds to normal mode with the embedding of triggers. Can only be changed when **FtStopped** is HIGH. This bit is cleared on reset. **0** Continuous formatting is disabled. **1** Continuous formatting is enabled.
[0]	EnFTC	Does not embed triggers into the formatted stream. TRACECTL indicates trace disable cycles and triggers, when it is present. Can only be changed when **FtStopped** is HIGH. This bit is cleared on reset. **0** Formatting is disabled. **1** Formatting is enabled.

3.11.11 Integration Test Miscellaneous Output Register 0

The ITMISCOP0 Register characteristics are:

Purpose Controls the values of some outputs from the ETB.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-89 shows the bit assignments.

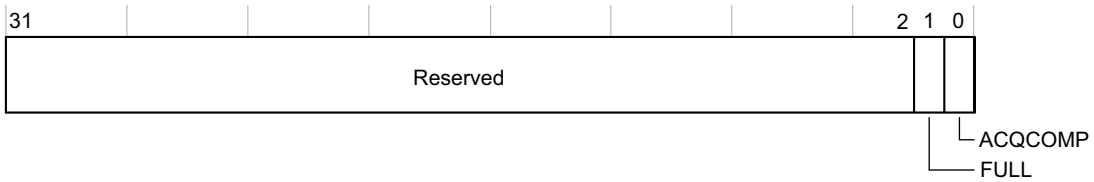


Figure 3-89 ITMISCOP0 Register bit assignments

Table 3-94 shows the bit assignments.

Table 3-94 ITMISCOP0 Register bit assignments

Bits	Name	Function
[31:2]	Reserved	-
[1]	FULL	Sets the value of **full** output. **0** Sets the value to 0. **1** Sets the value to 1.
[0]	ACQCOMP	Set the value of **acqcomp** output. **0** Set the value to 0. **1** Set the value to 1.

3.11.12 Integration Test Trigger In and Flush In Acknowledge Register

The ITTRFLINACK Register characteristics are:

Purpose Enables control of the **triginack** and **flushinack** outputs from the ETB.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-90 shows the bit assignments.

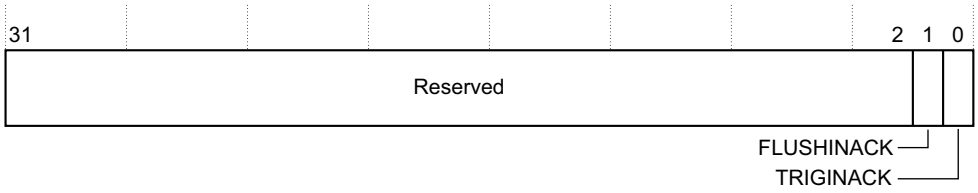


Figure 3-90 ITTRFLINACK Register bit assignments

Table 3-95 shows the bit assignments.

Table 3-95 ITTRFLINACK Register bit assignments

Bits	Name	Function
[31:2]	Reserved	-
[1]	FLUSHINACK	Set the value of **flushinack**. **0** Set the value of FLUSHINACK to 0. **1** Set the value of FLUSHINACK to 1.
[0]	TRIGINACK	Set the value of **triginack**. **0** Set the value of TRIGINACK to 0. **1** Set the value of TRIGINACK to 1.

3.11.13 Integration Test Trigger In and Flush In Register

The ITTRFLIN Register characteristics are:

Purpose Contains the values of the **flushin** and **trigin** inputs to the ETB.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-91 shows the bit assignments.

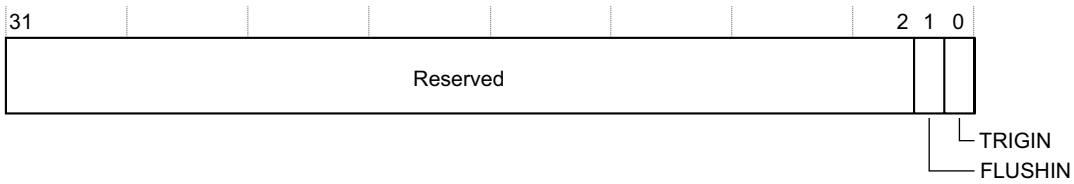


Figure 3-91 ITTRFLIN Register bit assignments

Table 3-96 shows the bit assignments.

Table 3-96 ITTRFLIN Register bit assignments

Bits	Name	Function
[31:2]	Reserved	-
[1]	FLUSHIN	Read the value of **flushin**. **0** **flushin** is LOW. **1** **flushin** is HIGH.
[0]	TRIGIN	Read the value of **trigin**. **0** **trigin** is LOW. **1** **trigin** is HIGH.

3.11.14 Integration Test ATB Data Register 0

The ITATBDATA0 Register characteristics are:

Purpose Contains the value of the **atdatas** inputs to the ETB. The values are only valid when **atvalids** is HIGH.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-92 shows the bit assignments.

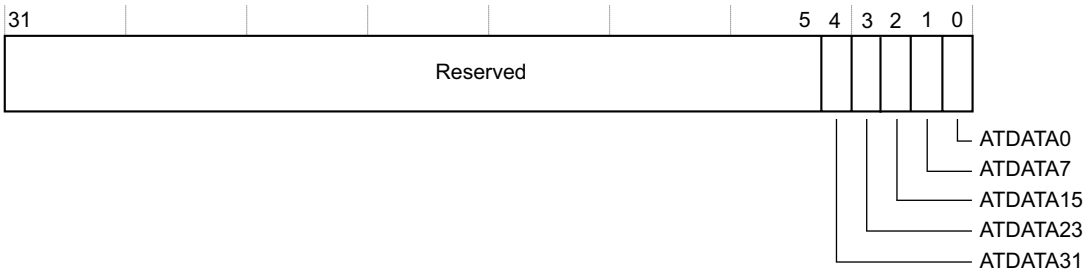


Figure 3-92 ITATBDATA0 Register bit assignments

Table 3-97 shows the bit assignments.

Table 3-97 ITATBDATA0 Register bit assignments

Bits	Name	Function
[31:5]	Reserved	-
[4]	ATDATA_31	Read the value of **atdatas[31]**. **0** **atdatas[31]** is 0. **1** **atdatas[31]** is 1.
[3]	ATDATA_23	Read the value of **atdatas[23]**. **0** **atdatas[23]** is 0. **1** **atdatas[23]** is 1.

Table 3-97 ITATBDATA0 Register bit assignments (continued)

Bits	Name	Function
[2]	ATDATA_15	Read the value of **atdatas[15]**. **0** **atdatas[15]** is 0. **1** **atdatas[15]** is 1.
[1]	ATDATA_7	Read the value of **atdatas[7]**. **0** **atdatas[7]** is 0. **1** **atdatas[7]** is 1.
[0]	ATDATA_0	Read the value of **atdatas[0]**. **0** **atdatas[0]** is 0. **1** **atdatas[0]** is 1.

3.11.15 Integration Test ATB Control Register 2

The ITATBCTR2 Register characteristics are:

Purpose Enables control of the **atreadys** and **afvalids** outputs of the ETB.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-93 shows the bit assignments.

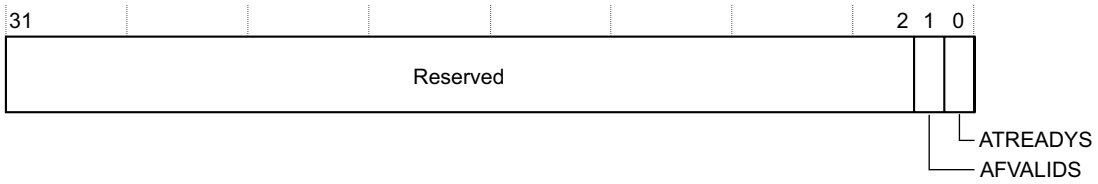


Figure 3-93 ITATBCTR2 Register bit assignments

Table 3-98 shows the bit assignments.

Table 3-98 ITATBCTR2 Register bit assignments

Bits	Name	Function
[31:2]	Reserved	-
[1]	AFVALIDS	Set the value of **afvalids**. **0** Set the value of **afvalids** to 0. **1** Set the value of **afvalids** to 1.
[0]	ATREADYS	Set the value of **atreadys**. **0** Set the value of **atreadys** to 0. **1** Set the value of **atreadys** to 1.

3.11.16 Integration Test ATB Control Register 1

The ITATBCTR1 Register characteristics are:

Purpose Contains the value of the **atids** input to the ETB. This value is valid only when **atvalids** is HIGH.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-94 shows the bit assignments.

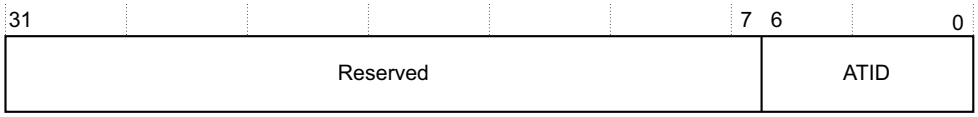


Figure 3-94 ITATBCTR1 Register bit assignments

Table 3-99 shows the bit assignments.

Table 3-99 ITATBCTR1 Register bit assignments

Bits	Name	Function
[31:7]	Reserved	-
[6:0]	ATID	Read the value of **atids**.

3.11.17 Integration Test ATB Control Register 0

The ITATBCTR0 Register characteristics are:

Purpose Captures the values of the **atvalids**, **afreadys**, and **atbytess** inputs to the ETB. To ensure the integration registers work correctly in a system, the value of **atbytess** is valid only when **atvalids**, bit[0], is HIGH.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-95 shows the bit assignments.

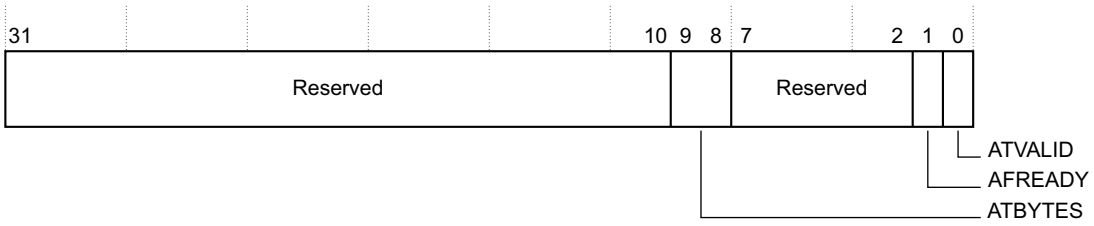


Figure 3-95 ITATBCTR0 Register bit assignments

Table 3-100 shows the bit assignments.

Table 3-100 ITATBCTR0 Register bit assignments

Bits	Name	Function
[31:10]	Reserved	-
[9:8]	ATBYTES	Read the value of **atbytess**.
[7:2]	Reserved	-
[1]	AFREADY	Read the value of **afreadys**. **0** **afreadys** is 0. **1** **afreadys** is 1.
[0]	ATVALID	Read the value of **atvalids**. **0** **atvalids** is 0. **1** **atvalids** is 1.

3.11.18 Integration Mode Control Register

The ITCTRL Register characteristics are:

Purpose Enables topology detection. For more information, see the *CoreSight Architecture Specification*. 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 the purposes of integration testing and topology detection.

Note

When a device is in integration mode, the intended functionality might not be available.

After performing integration or topology detection, you must reset the system to ensure correct behavior of CoreSight and other connected system components that the integration or topology detection can affect.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-96 shows the bit assignments.

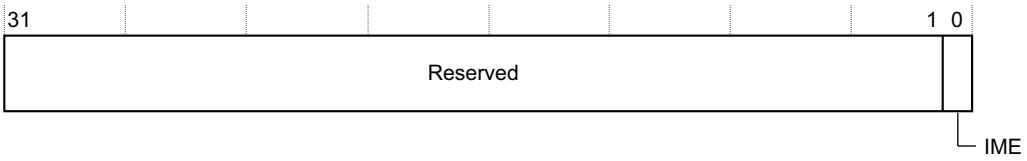


Figure 3-96 ITCTRL Register bit assignments

Table 3-101 shows the bit assignments.

Table 3-101 ITCTRL Register bit assignments

Bits	Name	Function
[31:1]	Reserved	-
[0]	IME	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. **0** Disable integration mode. **1** Enable integration mode.

3.11.19 Claim Tag Set Register

The CLAIMSET Register characteristics are:

Purpose Software can use the claim tag to coordinate application and debugger access to trace unit functionality. The claim tags have no effect on the operation of the component. The CLAIMSET Register sets bits in the claim tag, and determines the number of claim bits implemented.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-97 shows the bit assignments.

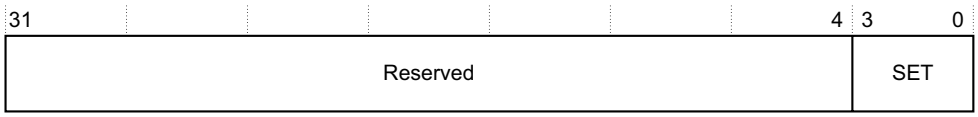


Figure 3-97 CLAIMSET Register bit assignments

Table 3-102 shows the bit assignments.

Table 3-102 CLAIMSET Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	SET_W	When you write 0 to bit[n] of this register, there is no effect on the claim tag bit[n]. When you write 1 to bit[n] of this register, bit[n] in the claim tag is set to 1.
[3:0]	SET_R	When you read 0 on a particular bit of this register, it indicates that this claim tag bit is not implemented. When you read 1 on a particular bit of this register, it indicates that this claim tag bit is implemented. `0b1111` Indicates that four bits of claim tag are implemented.

3.11.20 Claim Tag Clear Register

The CLAIMCLR Register characteristics are:

Purpose Software can use the claim tag to coordinate application and debugger access to trace unit functionality. The claim tags have no effect on the operation of the component. The CLAIMCLR Register clears the bits in the claim tag, and determines the current value of the claim tag.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-98 shows the bit assignments.

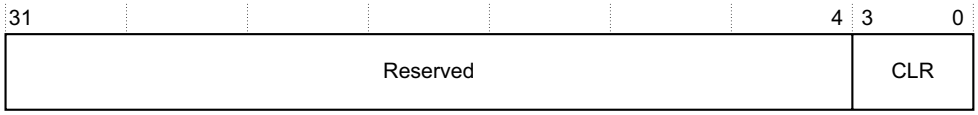


Figure 3-98 CLAIMCLR Register bit assignments

Table 3-103 shows the bit assignments.

Table 3-103 CLAIMCLR Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	CLR_W	When you write 0 to bit[n] of this register, there is no effect on the claim tag bit[n]. When you write 1 to bit[n] of this register, bit[n] in the claim tag is cleared to 0.
[3:0]	CLR_R	The value present reflects the present value of the Claim Tag.

3.11.21 Lock Access Register

The LAR characteristics are:

Purpose Controls write access from self-hosted, on-chip accesses. The LAR does not affect the accesses using the external debugger interface.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-99 shows the bit assignments.

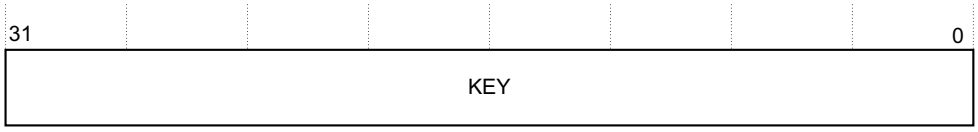


Figure 3-99 LAR bit assignments

Table 3-104 shows the bit assignments.

Table 3-104 LAR bit assignments

Bits	Name	Function
[31:0]	KEY	When you write `0xC5ACCE55`, subsequent write operations to this device are enabled. Any other value disables subsequent write operations.

3.11.22 Lock Status Register

The LSR characteristics are:

Purpose Indicates the status of the lock control mechanism. This lock prevents accidental writes by code being debugged. When locked, write accesses are denied for all registers except for the LAR. The lock registers do not affect accesses from the external debug interface. This register reads as 0 when accessed from the external debug interface.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-100 shows the bit assignments.

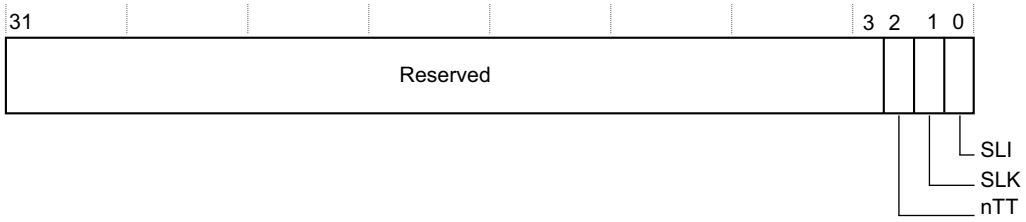


Figure 3-100 LSR bit assignments

Table 3-105 shows the bit assignments.

Table 3-105 LSR bit assignments

Bits	Name	Function
[31:3]	Reserved	-
[2]	nTT	Indicates that the LAR is implemented as 32-bit. `0b0` This component implements a 32-bit LAR.
[1]	SLK	Returns the present lock status of the device. `0b0` Indicates that write operations are permitted in this device. `0b1` Indicates that write operations are not permitted in this device. Read operations are permitted.
[0]	SLI	Indicates that a lock control mechanism is present in this device. `0b0` Indicates that a lock control mechanism is not present in this device. Write operations to the LAR Register are ignored. `0b1` Indicates that a lock control mechanism is present in this device.

3.11.23 Authentication Status Register

The AUTHSTATUS Register characteristics are:

Purpose Reports the required security level and present status.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-101 on page 3-95 shows the bit assignments.

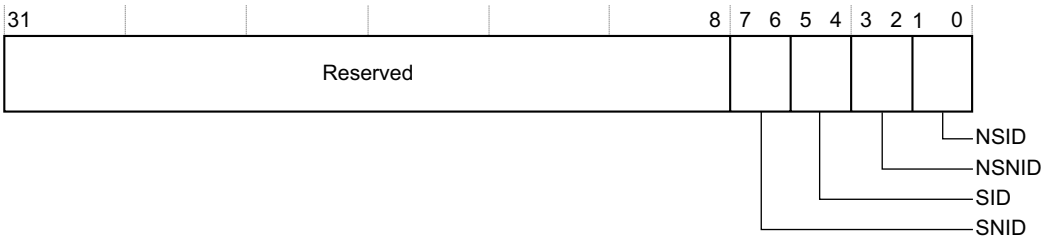


Figure 3-101 AUTHSTATUS Register bit assignments

Table 3-106 shows the bit assignments.

Table 3-106 AUTHSTATUS Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:6]	SNID	Indicates the security level for secure non-invasive debug: 0b00 Functionality is not implemented or is controlled elsewhere.
[5:4]	SID	Indicates the security level for secure invasive debug: 0b00 Functionality is not implemented or is controlled elsewhere.
[3:2]	NSNID	Indicates the security level for non-secure non-invasive debug: 0b00 Functionality is not implemented or is controlled elsewhere.
[1:0]	NSID	Indicates the security level for non-secure invasive debug: 0b00 Functionality is not implemented or is controlled elsewhere.

3.11.24 Device Configuration Register

The DEVID Register characteristics are:

Purpose Indicates the capabilities of the component.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-102 shows the bit assignments.

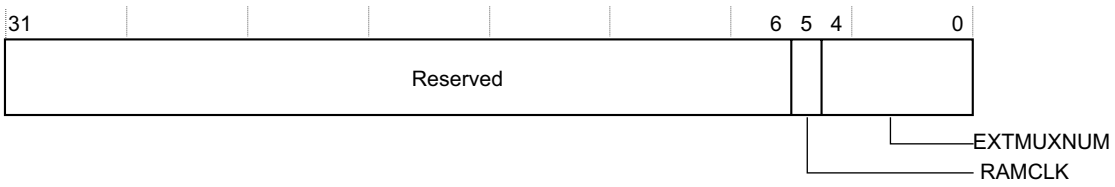


Figure 3-102 DEVID Register bit assignments

Table 3-107 shows the DEVID Register bit assignments.

Table 3-107 DEVID Register bit assignments

Bits	Name	Function
[31:6]	Reserved	-
[5]	RAMCLK	This bit returns 0 on reads to indicate that the ETB RAM operates synchronously to **atclk**. **0** The ETB RAM operates synchronously to **atclk**.
[4:0]	EXTMUXNUM	When non-zero, this value indicates the type or number of ATB multiplexing present on the input to the ATB. `0b0000` Only `0x00` is supported, that is, no multiplexing is present. This value is used to assist topology detection of the ATB structure.

3.11.25 Device Type Identifier Register

The DEVTYPE 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 See the register summary in Table 3-83 on page 3-76.

Figure 3-103 shows the bit assignments.

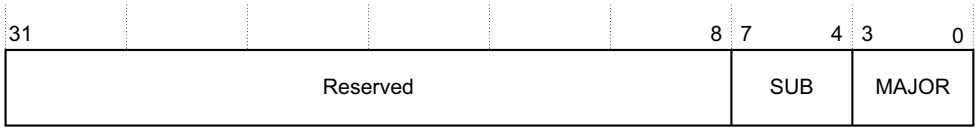


Figure 3-103 DEVTYPE Register bit assignments

Table 3-108 shows the bit assignments.

Table 3-108 DEVTYPE Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	SUB	Sub-classification of the type of the debug component as specified in the *CoreSight Architecture Specification* within the major classification as specified in the MAJOR field. `0b0010` This component is a trace buffer, ETB.
[3:0]	MAJOR	Major classification of the type of the debug component as specified in the *CoreSight Architecture Specification* for this debug and trace component. `0b0001` This component is a trace sink component.

3.11.26 Peripheral ID4 Register

The PIDR4 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-104 shows the bit assignments.

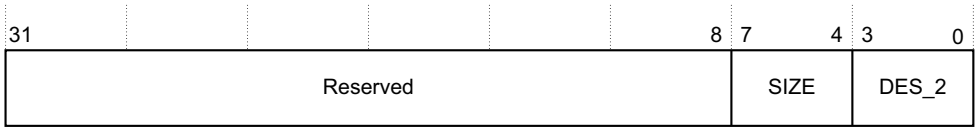


Figure 3-104 PIDR4 bit assignments

Table 3-109 shows the bit assignments.

Table 3-109 PIDR4 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	SIZE	This is a 4-bit value that indicates the total contiguous size of the memory window this component uses in powers of 2 from the standard 4KB. `0b0000` Indicates that the device only occupies 4KB of memory.
[3:0]	DES_2	This is the JEDEC JEP106 continuation code. This code, along with bits[6:4] of the identity code defined in the PIDR0 and bits[3:0] of the identity code defined in the PIDR1, identifies the designer of the component. `0b0100` JEDEC continuation code.

3.11.27 Peripheral ID5-7 registers

The PIDR5-7 characteristics are:

Purpose Reserved.

Usage constraints There are no usage constraints.

Configurations These registers are available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-105 shows the bit assignments.

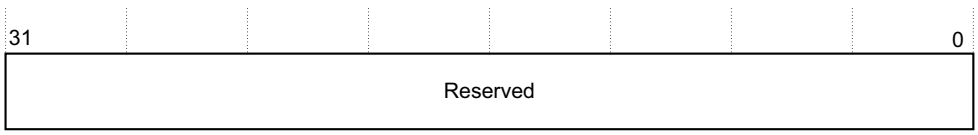


Figure 3-105 PIDR5-7 bit assignments

Table 3-110 shows the bit assignments.

Table 3-110 PIDR5-7 bit assignments

Bits	Name	Function
[31:0]	Reserved	-

3.11.28 Peripheral ID0 Register

The PIDR0 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 See the register summary in Table 3-83 on page 3-76.

Figure 3-106 shows the bit assignments.

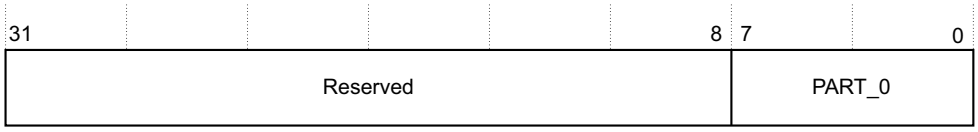


Figure 3-106 PIDR0 bit assignments

Table 3-111 shows the bit assignments.

Table 3-111 PIDR0 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PART_0	Bits[7:0] of the 12-bit part number of the component. The designer of the component assigns this part number. `0x07` Indicates bits[7:0] of the part number of the component.

3.11.29 Peripheral ID1 Register

The PIDR1 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 See the register summary in Table 3-83 on page 3-76.

Figure 3-107 on page 3-99 shows the bit assignments.

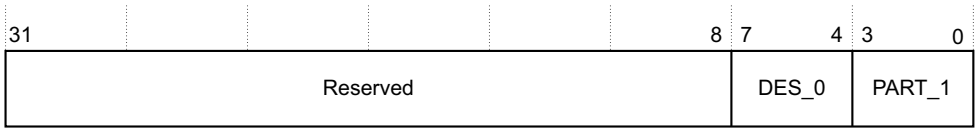


Figure 3-107 PIDR1 bit assignments

Table 3-112 shows the PIDR1 bit assignments.

Table 3-112 PIDR1 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	DES_0	Bits[3:0] of the JEDEC JEP106 identity code. This code, along with bits[6:4] of the identity code defined in the PIDR2 and the continuation code defined in the PIDR4, identifies the designer of the component. `0b1011` Bits[3:0] of the JEDEC JEP106 Identity Code. The default value is ARM.
[3:0]	PART_1	Bits[11:8] of the 12-bit part number of the component. The designer of the component assigns this part number. `0b1001` Indicates bits[11:8] of the part number of the component.

3.11.30 Peripheral ID2 Register

The PIDR2 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 See the register summary in Table 3-83 on page 3-76.

Figure 3-108 shows the bit assignments.

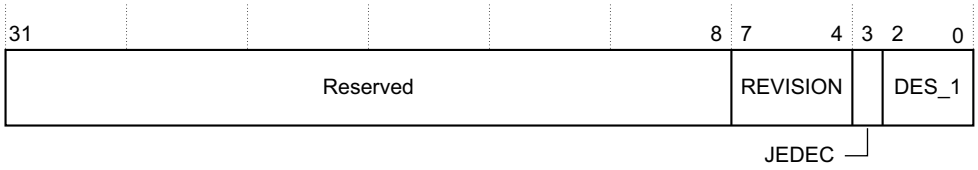


Figure 3-108 PIDR2 bit assignments

Table 3-113 shows the bit assignments.

Table 3-113 PIDR2 bit assignments

Bits	Name	Function
[31:8]	Reserved	-

Table 3-113 PIDR2 bit assignments (continued)

Bits	Name	Function
[7:4]	REVISION	An incremental value starting from 0b0000 for the first revision of this component. This increases by 1 for both major and minor revisions and is used to identify the major or minor revisions. 0b0011 This device is at r0p3.
[3]	JEDEC	Always set. Indicates whether the JEDEC assigned designer ID is used. 0b1 Indicates that a JEDEC assigned designer ID is used.
[2:0]	DES_1	Bits[6:4] of the JEDEC JEP106 identity code. This code, along with bits[3:0] of the identity code defined in the PIDR1 and the continuation code defined in the PIDR4 Register, identifies the designer of the component. 0b011 Bits[6:4] of the JEDEC JEP106 Identity Code. The default value is ARM.

3.11.31 Peripheral ID3 Register

The PIDR3 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-109 shows the bit assignments.

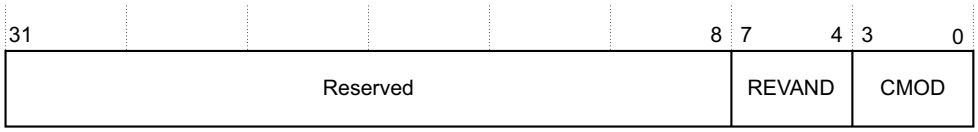


Figure 3-109 PIDR3 bit assignments

Table 3-114 shows the bit assignments.

Table 3-114 PIDR3 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	REVAND	Indicates minor errata fixes specific to the revision of the component being used, for example metal fixes after implementation. In most cases, this field is zero. ARM recommends that the component designers ensure that a metal fix can change this field if required, for example, by driving it from registers that reset to zero. 0b0000 Indicates that there are no errata fixes to this component.
[3:0]	CMOD	Indicates whether the customer has modified the behavior of the component. In most cases, this field is zero. The customer changes this value on modifications to this component. 0b0000 Indicates that the customer has not modified this component.

3.11.32 Component ID0 Register

The CIDR0 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-110 shows the bit assignments.

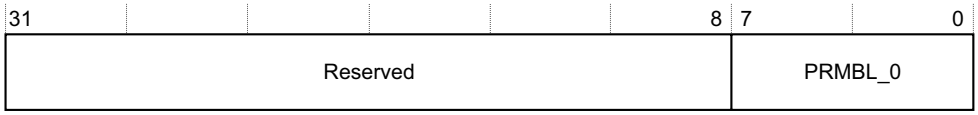


Figure 3-110 CIDR0 bit assignments

Table 3-115 shows the bit assignments.

Table 3-115 CIDR0 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_0	Contains bits[7:0] of the component identification code. `0x0D` Bits[7:0] of the identification code.

3.11.33 Component ID1 Register

The CIDR1 characteristics are:

Purpose A component identification register that indicates 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 See the register summary in Table 3-83 on page 3-76.

Figure 3-111 shows the bit assignments.

Figure 3-111 CIDR1 bit assignments

Table 3-116 shows the bit assignments.

Table 3-116 CIDR1 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	CLASS	Class of the component, for example, if the component is a ROM table or a generic CoreSight component. Contains bits[15:12] of the component identification code. `0b1001` Indicates that the component is a CoreSight component.
[3:0]	PRMBL_1	Contains bits[11:8] of the component identification code. `0b0000` Bits[11:8] of the identification code.

3.11.34 Component ID2 Register

The CIDR2 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-112 shows the bit assignments.

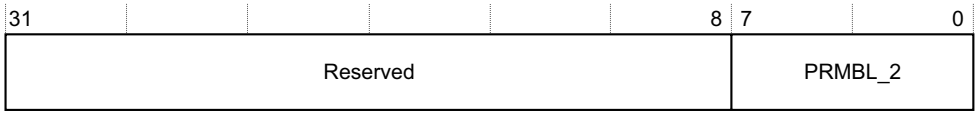


Figure 3-112 CIDR2 bit assignments

Table 3-117 shows the bit assignments.

Table 3-117 CIDR2 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_2	Contains bits[23:16] of the component identification code. 0x05 Bits[23:16] of the identification code.

3.11.35 Component ID3 Register

The CIDR3 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-83 on page 3-76.

Figure 3-113 shows the bit assignments.

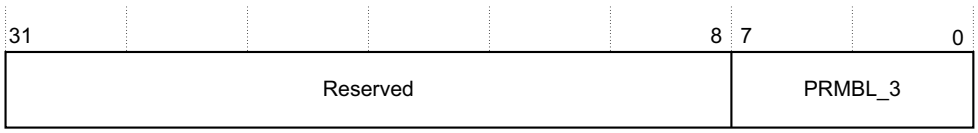


Figure 3-113 CIDR3 bit assignments

Table 3-118 shows the bit assignments.

Table 3-118 CIDR3 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_3	Contains bits[31:24] of the component identification code. `0xB1` Bits[31:24] of the identification code.

3.12 CTI register summary

Table 3-119 shows the CTI registers in offset order from the base memory address.

Table 3-119 CTI register summary

Offset	Name	Type	Reset	Description
0x000	CTICONTROL	RW	0x00000000	*CTI Control Register* on page 3-106
0x010	CTIINTACK	WO	0x00000000	*CTI Interrupt Acknowledge Register* on page 3-106
0x014	CTIAPPSET	RW	0x00000000	*CTI Application Trigger Set Register* on page 3-107
0x018	CTIAPPCLEAR	WO	0x00000000	*CTI Application Trigger Clear Register* on page 3-108
0x01C	CTIAPPPULSE	WO	0x00000000	*CTI Application Pulse Register* on page 3-108
0x020	CTIINEN0	RW	0x00000000	*CTI Trigger 0 to Channel Enable Register* on page 3-109
0x024	CTIINEN1	RW	0x00000000	*CTI Trigger 1 to Channel Enable Register* on page 3-109
0x028	CTIINEN2	RW	0x00000000	*CTI Trigger 2 to Channel Enable Register* on page 3-110
0x02C	CTIINEN3	RW	0x00000000	*CTI Trigger 3 to Channel Enable Register* on page 3-111
0x030	CTIINEN4	RW	0x00000000	*CTI Trigger 4 to Channel Enable Register* on page 3-112
0x034	CTIINEN5	RW	0x00000000	*CTI Trigger 5 to Channel Enable Register* on page 3-112
0x038	CTIINEN6	RW	0x00000000	*CTI Trigger 6 to Channel Enable Register* on page 3-113
0x03C	CTIINEN7	RW	0x00000000	*CTI Trigger 7 to Channel Enable Register* on page 3-114
0x0A0	CTIOUTEN0	RW	0x00000000	*CTI Channel to Trigger 0 Enable Register* on page 3-115
0x0A4	CTIOUTEN1	RW	0x00000000	*CTI Channel to Trigger 1 Enable Register* on page 3-115
0x0A8	CTIOUTEN2	RW	0x00000000	*CTI Channel to Trigger 2 Enable Register* on page 3-116
0x0AC	CTIOUTEN3	RW	0x00000000	*CTI Channel to Trigger 3 Enable Register* on page 3-117
0x0B0	CTIOUTEN4	RW	0x00000000	*CTI Channel to Trigger 4 Enable Register* on page 3-117
0x0B4	CTIOUTEN5	RW	0x00000000	*CTI Channel to Trigger 5 Enable Register* on page 3-118
0x0B8	CTIOUTEN6	RW	0x00000000	*CTI Channel to Trigger 6 Enable Register* on page 3-119
0x0BC	CTIOUTEN7	RW	0x00000000	*CTI Channel to Trigger 7 Enable Register* on page 3-120
0x130	CTITRIGINSTATUS	RO	0x00000000	*CTI Trigger In Status Register* on page 3-120
0x134	CTITRIGOUTSTATUS	RO	0x00000000	*CTI Trigger Out Status Register* on page 3-121
0x138	CTICHINSTATUS	RO	0x00000000	*CTI Channel In Status Register* on page 3-121
0x13C	CTICHOUTSTATUS	RO	0x00000000	*CTI Channel Out Status Register* on page 3-122
0x140	CTIGATE	RW	0x0000000F	*Enable CTI Channel Gate Register* on page 3-123
0x144	asicctl	RW	0x00000000	*External Multiplexer Control Register* on page 3-123
0xEDC	ITCHINACK	WO	0x00000000	*Integration Test Channel Input Acknowledge Register* on page 3-124
0xEE0	ITTRIGINACK	WO	0x00000000	*Integration Test Trigger Input Acknowledge Register* on page 3-124

Table 3-119 CTI register summary (continued)

Offset	Name	Type	Reset	Description
0xEE4	ITCHOUT	WO	0x00000000	*Integration Test Channel Output Register* on page 3-125
0xEE8	ITTRIGOUT	WO	0x00000000	*Integration Test Trigger Output Register* on page 3-125
0xEEC	ITCHOUTACK	RO	0x00000000	*Integration Test Channel Output Acknowledge Register* on page 3-126
0xEF0	ITTRIGOUTACK	RO	0x00000000	*Integration Test Trigger Output Acknowledge Register* on page 3-126
0xEF4	ITCHIN	RO	0x00000000	*Integration Test Channel Input Register* on page 3-127
0xEF8	ITTRIGIN	RO	0x00000000	*Integration Test Trigger Input Register* on page 3-127
0xF00	ITCTRL	RW	0x00000000	*Integration Mode Control Register* on page 3-128
0xFA0	CLAIMSET	RW	0x0000000F	*Claim Tag Set Register* on page 3-129
0xFA4	CLAIMCLR	RW	0x00000000	*Claim Tag Clear Register* on page 3-129
0xFB0	LAR	WO	0x00000000	*Lock Access Register* on page 3-130
0xFB4	LSR	RO	0x00000003	*Lock Status Register* on page 3-131
0xFB8	AUTHSTATUS	RO	0x00000005	*Authentication Status Register* on page 3-131
0xFC8	DEVID	RO	0x00040800	*Device Configuration Register* on page 3-132
0xFCC	DEVTYPE	RO	0x00000014	*Device Type Identifier Register* on page 3-133
0xFD0	PIDR4	RO	0x00000004	*Peripheral ID4 Register* on page 3-134
0xFD4	PIDR5	RO	0x00000000	*Peripheral ID5-7 Registers* on page 3-134
0xFD8	PIDR6	RO	0x00000000	
0xFDC	PIDR7	RO	0x00000000	
0xFE0	PIDR0	RO	0x00000006	*Peripheral ID0 Register* on page 3-135
0xFE4	PIDR1	RO	0x000000B9	*Peripheral ID1 Register* on page 3-135
0xFE8	PIDR2	RO	0x0000004B	*Peripheral ID2 Register* on page 3-136
0xFEC	PIDR3	RO	0x00000000	*Peripheral ID3 Register* on page 3-137
0xFF0	CIDR0	RO	0x0000000D	*Component ID0 Register* on page 3-137
0xFF4	CIDR1	RO	0x00000090	*Component ID1 Register* on page 3-138
0xFF8	CIDR2	RO	0x00000005	*Component ID2 Register* on page 3-139
0xFFC	CIDR3	RO	0x000000B1	*Component ID3 Register* on page 3-139

3.13 CTI register descriptions

This section describes the CTI registers. Table 3-119 on page 3-104 provides cross references to individual registers.

3.13.1 CTI Control Register

The CTICONTROL Register characteristics are:

Purpose Enables the CTI.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-114 shows the bit assignments.

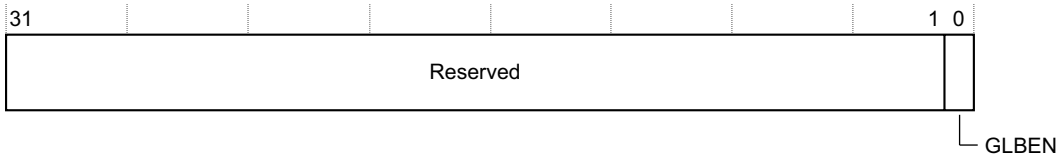


Figure 3-114 CTICONTROL Register bit assignments

Table 3-120 shows the bit assignments.

Table 3-120 CTICONTROL Register bit assignments

Bits	Name	Function
[31:1]	Reserved	-
[0]	GLBEN	Enables or disables the ECT. **0** When this bit is 0, all cross triggering mapping logic functionality is disabled. **1** When this bit is 1, cross triggering mapping logic functionality is enabled.

3.13.2 CTI Interrupt Acknowledge Register

The CTIINTACK Register characteristics are:

Purpose Any bits written as a 1 cause the **ctitrigout** output signal to be acknowledged. The acknowledgement is cleared when **MAPTRIGOUT** is deactivated. This register is used when the **ctitrigout** is used as a sticky output, that is, no hardware acknowledge is supplied, and a software acknowledge is required.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-115 on page 3-107 shows the bit assignments.

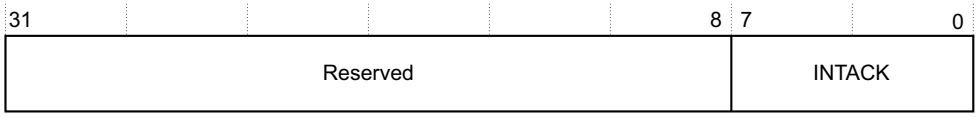


Figure 3-115 CTIINTACK Register bit assignments

Table 3-121 shows the bit assignments.

Table 3-121 CTIINTACK Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	INTACK	Acknowledges the corresponding **ctitrigout** output. There is one bit of the register for each **ctitrigout** output. When a 1 is written to a bit in this register, the corresponding **ctitrigout** is acknowledged and is cleared when **MAPTRIGOUT** is LOW. Writing a 0 to any of the bits in this register has no effect.

3.13.3 CTI Application Trigger Set Register

The CTIAPPSET Register characteristics are:

Purpose A write to this register causes a channel event to be raised, corresponding to the bit written to.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-116 shows the bit assignments.

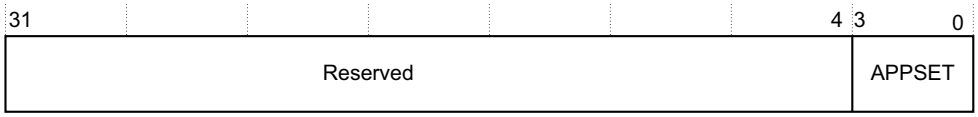


Figure 3-116 CTIAPPSET Register bit assignments

Table 3-122 shows the bit assignments.

Table 3-122 CTIAPPSET Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	APPSET	Setting a bit HIGH generates a channel event for the selected channel. There is one bit of the register for each channel. Reads as follows: **0** Application trigger is inactive. **1** Application trigger is active. Writes as follows: **0** No effect. **1** Generate channel event.

3.13.4 CTI Application Trigger Clear Register

The CTIAPPCLEAR Register characteristics are:

Purpose A write to this register causes a channel event to be cleared, corresponding to the bit written to.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-117 shows the bit assignments.

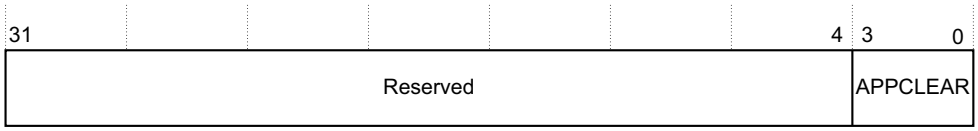


Figure 3-117 CTIAPPCLEAR Register bit assignments

Table 3-123 shows the bit assignments.

Table 3-123 CTIAPPCLEAR Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	APPCLEAR	Clears corresponding bits in the CTIAPPSET Register. There is one bit of the register for each channel. When a 1 is written to a bit in this register, the corresponding application trigger is disabled in the CTIAPPSET Register. Writing a 0 to any of the bits in this register has no effect.

3.13.5 CTI Application Pulse Register

The CTIAPPPULSE Register characteristics are:

Purpose A write to this register causes a channel event pulse, one **cticlk** period, to be generated, corresponding to the bit written to. The pulse external to the ECT can be extended to multi-cycle by the handshaking interface circuits. This register clears itself immediately, so it can be repeatedly written to without software having to clear it.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-118 shows the bit assignments.

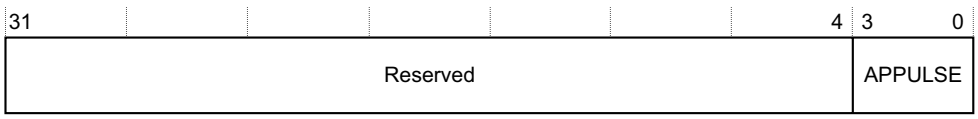


Figure 3-118 CTIAPPPULSE Register bit assignments

Table 3-124 shows the bit assignments.

Table 3-124 CTIAPPPULSE Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	APPULSE	Setting a bit HIGH generates a channel event pulse for the selected channel. There is one bit of the register for each channel. When a 1 is written to a bit in this register, a corresponding channel event pulse is generated for one **cticlk** period. Writing a 0 to any of the bits in this register has no effect.

3.13.6 CTI Trigger 0 to Channel Enable Register

The CTIINEN0 Register characteristics are:

Purpose Enables the signaling of an event on CTM channels when the core issues a trigger, **ctitrigin**, to the CTI. There is a bit for each of the four channels implemented. This register does not affect the application trigger operations.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-119 shows the bit assignments.

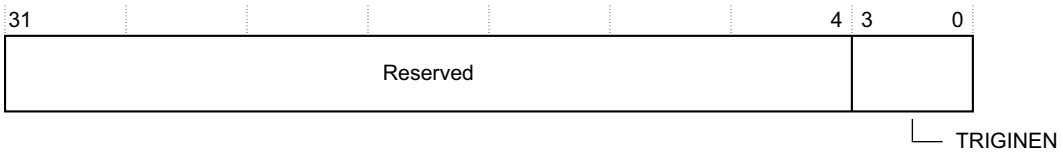


Figure 3-119 CTIINEN0 Register bit assignments

Table 3-125 shows the it assignments.

Table 3-125 CTIINEN0 Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	TRIGINEN	Enables a cross trigger event to the corresponding channel when a **ctitrigin** input is activated. There is one bit of the field for each of the four channels.When a 1 is written to a bit in this register, it enables the **ctitrigin** signal to generate an event on the respective channel of the CTM. For example, **triginen[0]** set to 1 enables **ctitrigin[0]** onto channel 0. Writing a 0 to any of the bits in this register disables the **ctitrigin** signal from generating an event on the respective channel of the CTM. Reading this register returns the programmed value.

3.13.7 CTI Trigger 1 to Channel Enable Register

The CTIINEN1 Register characteristics are:

Purpose Enables the signaling of an event on CTM channels when the core issues a trigger, **ctitrigin**, to the CTI. There is a bit for each of the four channels implemented. This register does not affect the application trigger operations.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-120 shows the bit assignments.

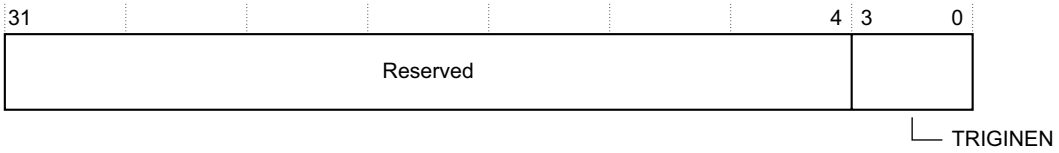


Figure 3-120 CTIINEN1 Register bit assignments

Table 3-126 shows the bit assignments.

Table 3-126 CTIINEN1 Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	TRIGINEN	Enables a cross trigger event to the corresponding channel when a **ctitrigin** is activated. There is one bit of the field for each of the four channels. When a 1 is written to a bit in this register, it enables the **ctitrigin** signal to generate an event on the respective channel of the CTM. For example, **triginen[0]** set to 1 enables **ctitrigin[1]** onto channel 0. Writing a 0 to any of the bits in this register disables the **ctitrigin** signal from generating an event on the respective channel of the CTM. Reading this register returns the programmed value.

3.13.8 CTI Trigger 2 to Channel Enable Register

The CTIINEN2 Register characteristics are:

Purpose Enables the signaling of an event on CTM channels when the core issues a trigger, **ctitrigin**, to the CTI. There is a bit for each of the four channels implemented. This register does not affect the application trigger operations.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-121 shows the assignments.

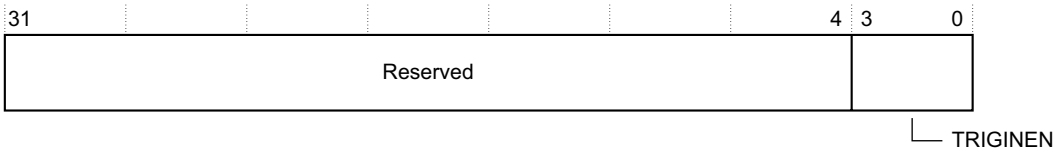


Figure 3-121 CTIINEN2 Register bit assignments

Table 3-127 shows the bit assignments.

Table 3-127 CTIINEN2 Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	TRIGINEN	Enables a cross trigger event to the corresponding channel when an **ctitrigin** is activated. There is one bit of the field for each of the four channels. When a 1 is written to a bit in this register, it enables the **ctitrigin** signal to generate an event on the respective channel of the CTM. For example, **triginen[0]** set to 1 enables **ctitrigin[2]** onto channel 0. Writing a 0 to any of the bits in this register disables the **ctitrigin** signal from generating an event on the respective channel of the CTM. Reading this register returns the programmed value.

3.13.9 CTI Trigger 3 to Channel Enable Register

The CTIINEN3 Register characteristics are:

Purpose Enables the signaling of an event on CTM channels when the core issues a trigger, **ctitrigin**, to the CTI. There is a bit for each of the four channels implemented. This register does not affect the application trigger operations.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-122 shows the bit assignments.

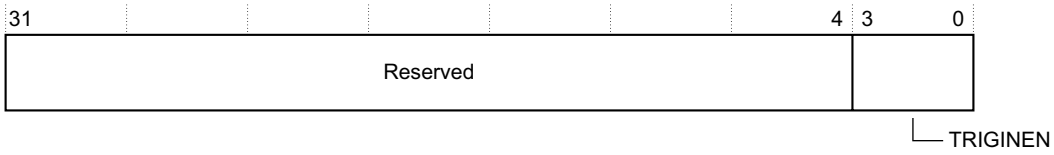


Figure 3-122 CTIINEN3 Register bit assignments

Table 3-128 shows the bit assignments.

Table 3-128 CTIINEN3 Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	TRIGINEN	Enables a cross trigger event to the corresponding channel when an **ctitrigin** is activated. There is one bit of the field for each of the four channels. When a 1 is written to a bit in this register, it enables the **ctitrigin** signal to generate an event on the respective channel of the CTM. For example, **triginen[0]** set to 1 enables **ctitrigin[3]** onto channel 0. Writing a 0 to any of the bits in this register disables the **ctitrigin** signal from generating an event on the respective channel of the CTM. Reading this register returns the programmed value.

3.13.10 CTI Trigger 4 to Channel Enable Register

The CTIINEN4 Register characteristics are:

Purpose Enables the signaling of an event on CTM channels when the core issues a trigger, **ctitrigin**, to the CTI. There is a bit for each of the four channels implemented. This register does not affect the application trigger operations.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-123 shows the CTIINEN4 Register bit assignments.

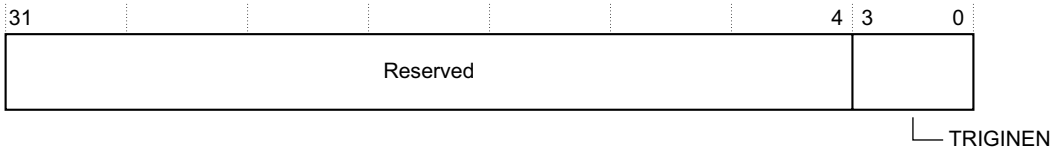


Figure 3-123 CTIINEN4 Register bit assignments

Table 3-129 shows the CTIINEN4 Register bit assignments.

Table 3-129 CTIINEN4 Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	TRIGINEN	Enables a cross trigger event to the corresponding channel when an **ctitrigin** is activated. There is one bit of the field for each of the four channels. When a 1 is written to a bit in this register, it enables the **ctitrigin** signal to generate an event on the respective channel of the CTM. For example, **triginen[0]** set to 1 enables **ctitrigin[4]** onto channel 0. Writing a 0 to any of the bits in this register disables the **ctitrigin** signal from generating an event on the respective channel of the CTM. Reading this register returns the programmed value.

3.13.11 CTI Trigger 5 to Channel Enable Register

The CTIINEN5 Register characteristics are:

Purpose Enables the signaling of an event on CTM channels when the core issues a trigger, **ctitrigin**, to the CTI. There is a bit for each of the four channels implemented. This register does not affect the application trigger operations.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-124 on page 3-113 shows the bit assignments.

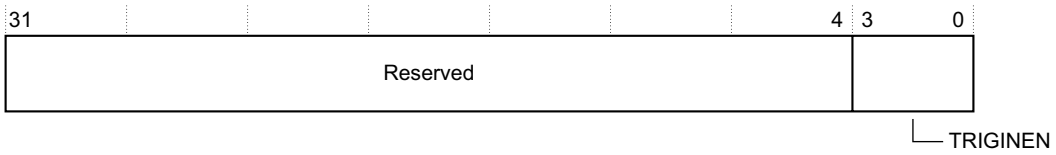


Figure 3-124 CTIINEN5 Register bit assignments

Table 3-130 shows the bit assignments.

Table 3-130 CTIINEN5 Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	TRIGINEN	Enables a cross trigger event to the corresponding channel when an **ctitrigin** is activated. There is one bit of the field for each of the four channels. When a 1 is written to a bit in this register, it enables the **ctitrigin** signal to generate an event on the respective channel of the CTM. For example, **triginen[0]** set to 1 enables **ctitrigin[5]** onto channel 0. Writing a 0 to any of the bits in this register disables the **ctitrigin** signal from generating an event on the respective channel of the CTM. Reading this register returns the programmed value.

3.13.12 CTI Trigger 6 to Channel Enable Register

The CTIINEN6 Register characteristics are:

Purpose Enables the signaling of an event on CTM channels when the core issues a trigger, **ctitrigin**, to the CTI. There is a bit for each of the four channels implemented. This register does not affect the application trigger operations.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-125 shows the bit assignments.

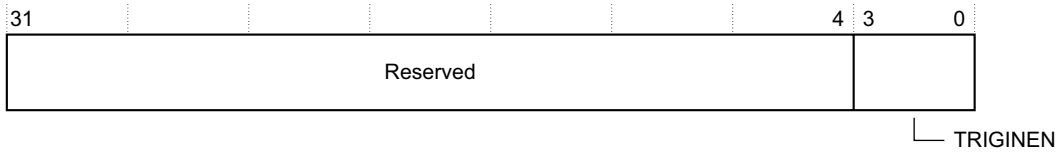


Figure 3-125 CTIINEN6 Register bit assignments

Table 3-131 shows the bit assignments.

Table 3-131 CTIINEN6 Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	TRIGINEN	Enables a cross trigger event to the corresponding channel when an **ctitrigin** is activated. There is one bit of the field for each of the four channels. When a 1 is written to a bit in this register, it enables the **ctitrigin** signal to generate an event on the respective channel of the CTM. For example, **triginen[0]** set to 1 enables **ctitrigin[6]** onto channel 0. Writing a 0 to any of the bits in this register disables the **ctitrigin** signal from generating an event on the respective channel of the CTM. Reading this register returns the programmed value.

3.13.13 CTI Trigger 7 to Channel Enable Register

The CTIINEN7 Register characteristics are:

Purpose Enables the signaling of an event on CTM channels when the core issues a trigger, **ctitrigin**, to the CTI. There is a bit for each of the four channels implemented. This register does not affect the application trigger operations.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-126 shows the bit assignments.

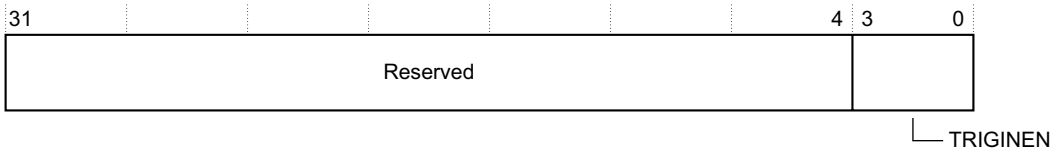


Figure 3-126 CTIINEN7 Register bit assignments

Table 3-132 shows the bit assignments.

Table 3-132 CTIINEN7 Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	TRIGINEN	Enables a cross trigger event to the corresponding channel when an **ctitrigin** is activated. There is one bit of the field for each of the four channels. When a 1 is written to a bit in this register, it enables the **ctitrigin** signal to generate an event on the respective channel of the CTM. For example, **triginen[0]** set to 1 enables **ctitrigin[7]** onto channel 0. Writing a 0 to any of the bits in this register disables the **ctitrigin** signal from generating an event on the respective channel of the CTM. Reading this register returns the programmed value.

3.13.14 CTI Channel to Trigger 0 Enable Register

The CTIOUTEN0 Register characteristics are:

Purpose Defines which channels can generate a **ctitrigout[0]** output. There is a bit for each of the four channels implemented. This register affects the mapping from application trigger to trigger outputs.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-127 shows the bit assignments.

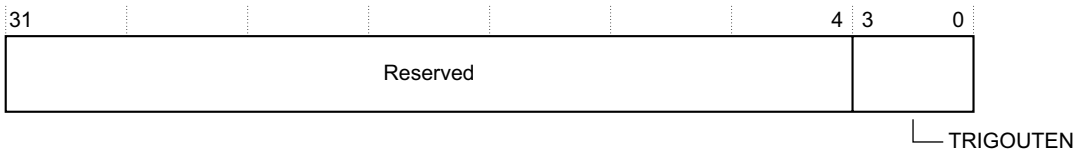


Figure 3-127 CTIOUTEN0 Register bit assignments

Table 3-133 shows the bit assignments.

Table 3-133 CTIOUTEN0 Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	TRIGOUTEN	Changing the value of this bit from a 0 to a 1 enables a channel event for the corresponding channel to generate a **ctitrigout[0]** output. There is one bit of the field for each of the four channels. When a 1 is written to a bit in this register, the channel input, **ctichin**, from the CTM is routed to the **ctitrigout** output. For example, enabling bit 0 enables **ctichin[0]** to cause a trigger event on the **ctitrigout[0]** output. When a 0 is written to any of the bits in this register, the channel input, **ctichin** from the CTM is not routed to the **ctitrigout** output. Reading this register returns the programmed value.

3.13.15 CTI Channel to Trigger 1 Enable Register

The CTIOUTEN1 Register characteristics are:

Purpose Defines which channels can generate a **ctitrigout[1]** output. There is a bit for each of the four channels implemented. This register affects the mapping from application trigger to trigger outputs.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-128 on page 3-116 shows the bit assignments.

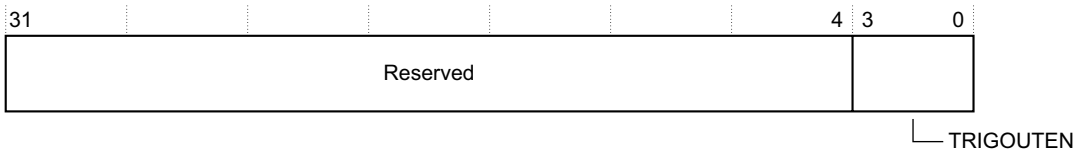


Figure 3-128 CTIOUTEN1 Register bit assignments

Table 3-134 shows the bit assignments.

Table 3-134 CTIOUTEN1 Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	TRIGOUTEN	Changing the value of this bit from a 0 to a 1 enables a channel event for the corresponding channel to generate a **ctitrigout[1]** output. There is one bit of the field for each of the four channels. When a 1 is written to a bit in this register, the channel input, **ctichin**, from the CTM is routed to the **ctitrigout** output. For example, enabling bit 0 enables **ctichin[0]** to cause a trigger event on the **ctitrigout[1]** output. When a 0 is written to any of the bits in this register, the channel input, **ctichin**, from the CTM is not routed to the **ctitrigout** output. Reading this register returns the programmed value.

3.13.16 CTI Channel to Trigger 2 Enable Register

The CTIOUTEN2 Register characteristics are:

Purpose Defines which channels can generate a **ctitrigout[2]** output. There is a bit for each of the four channels implemented. This register affects the mapping from application trigger to trigger outputs.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-129 shows the bit assignments.

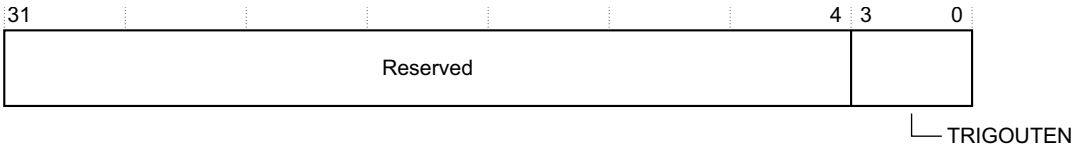


Figure 3-129 CTIOUTEN2 Register bit assignments

Table 3-135 shows the bit assignments.

Table 3-135 CTIOUTEN2 Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	TRIGOUTEN	Changing the value of this bit from a 0 to a 1 enables a channel event for the corresponding channel to generate a **ctitrigout[2]** output. There is one bit of the field for each of the four channels. When a 1 is written to a bit in this register, the channel input, **ctichin**, from the CTM is routed to the **ctitrigout** output. For example, enabling bit 0 enables **ctichin[0]** to cause a trigger event on the **ctitrigout[2]** output. When a 0 is written to any of the bits in this register, the channel input, **ctichin**, from the CTM is not routed to the **ctitrigout** output. Reading this register returns the programmed value.

3.13.17 CTI Channel to Trigger 3 Enable Register

The CTIOUTEN3 Register characteristics are:

Purpose Defines which channels can generate a **ctitrigout[3]** output. There is a bit for each of the four channels implemented. This register affects the mapping from application trigger to trigger outputs.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-130 shows the bit assignments.

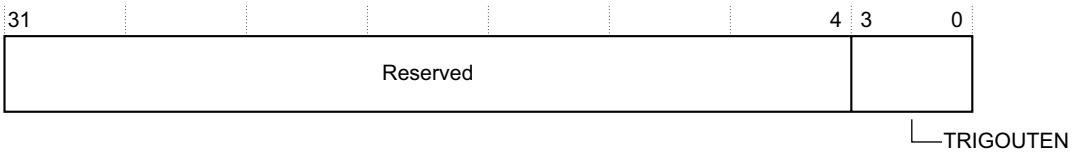


Figure 3-130 CTIOUTEN3 Register bit assignments

Table 3-136 shows the bit assignments.

Table 3-136 CTIOUTEN3 Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	TRIGOUTEN	Changing the value of this bit from a 0 to a 1 enables a channel event for the corresponding channel to generate a **ctitrigout[3]** output. There is one bit of the field for each of the four channels. When a 1 is written to a bit in this register, the channel input, **ctichin**, from the CTM is routed to the **ctitrigout** output. For example, enabling bit 0 enables **ctichin[0]** to cause a trigger event on the **ctitrigout[3]** output. When a 0 is written to any of the bits in this register, the channel input, **ctichin**, from the CTM is not routed to the **ctitrigout** output. Reading this register returns the programmed value.

3.13.18 CTI Channel to Trigger 4 Enable Register

The CTIOUTEN4 Register characteristics are:

Purpose Defines which channels can generate a **ctitrigout[4]** output. There is a bit for each of the four channels implemented. This register affects the mapping from application trigger to trigger outputs.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-131 on page 3-118 shows the bit assignments.

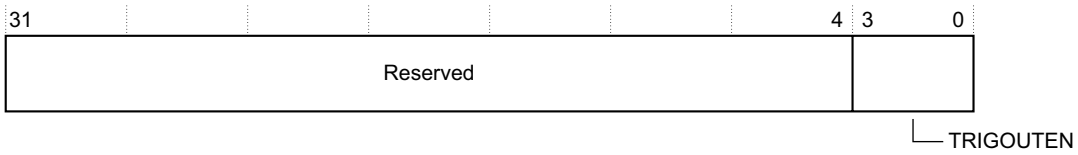


Figure 3-131 CTIOUTEN4 Register bit assignments

Table 3-137 shows the bit assignments.

Table 3-137 CTIOUTEN4 Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	TRIGOUTEN	Changing the value of this bit from a 0 to a 1 enables a channel event for the corresponding channel to generate a **ctitrigout[4]** output. There is one bit of the field for each of the four channels. When a 1 is written to a bit in this register, the channel input, **ctichin** from the CTM is routed to the **ctitrigout** output. For example, enabling bit 0 enables **ctichin[0]** to cause a trigger event on the **ctitrigout[4]** output. When a 0 is written to any of the bits in this register, the channel input, **ctichin**, from the CTM is not routed to the **ctitrigout** output. Reading this register returns the programmed value.

3.13.19 CTI Channel to Trigger 5 Enable Register

The CTIOUTEN5 Register characteristics are:

Purpose Define which channels can generate a **ctitrigout[5]** output. There is a bit for each of the four channels implemented. This register affects the mapping from application trigger to trigger outputs.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-132 shows the bit assignments.

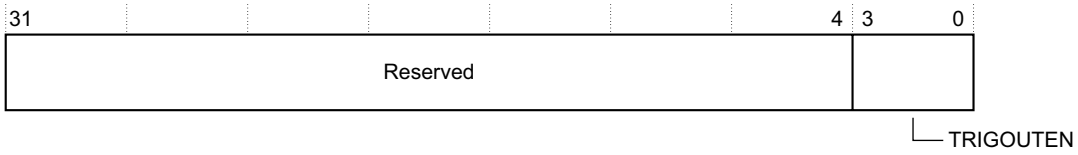


Figure 3-132 CTIOUTEN5 Register bit assignments

Table 3-138 shows the bit assignments.

Table 3-138 CTIOUTEN5 Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	TRIGOUTEN	Changing the value of this bit from a 0 to a 1 enables a channel event for the corresponding channel to generate a **ctitrigout[5]** output. There is one bit of the field for each of the four channels.When a 1 is written to a bit in this register, the channel input, **ctichin**, from the CTM is routed to the **ctitrigout** output. For example, enabling bit 0 enables **ctichin[0]** to cause a trigger event on the **ctitrigout[5]** output. When a 0 is written to any of the bits in this register, the channel input, **ctichin**, from the CTM is not routed to the **ctitrigout** output.Reading this register returns the programmed value.

3.13.20 CTI Channel to Trigger 6 Enable Register

The CTIOUTEN6 Register characteristics are:

Purpose Defines which channels can generate a **ctitrigout[6]** output. There is a bit for each of the four channels implemented. This register affects the mapping from application trigger to trigger outputs.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-133 shows the bit assignments.

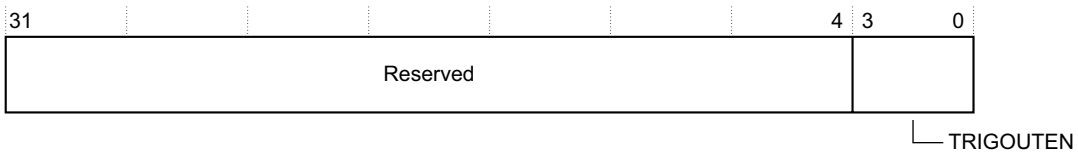


Figure 3-133 CTIOUTEN6 Register bit assignments

Table 3-139 shows the bit assignments.

Table 3-139 CTIOUTEN6 Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	TRIGOUTEN	Changing the value of this bit from a 0 to a 1 enables a channel event for the corresponding channel to generate a **ctitrigout[6]** output. There is one bit of the field for each of the four channels. When a 1 is written to a bit in this register, the channel input, **ctichin**, from the CTM is routed to the **ctitrigout** output. For example, enabling bit 0 enables **ctichin[0]** to cause a trigger event on the **ctitrigout[6]** output. When a 0 is written to any of the bits in this register, the channel input, **ctichin**, from the CTM is not routed to the **ctitrigout** output. Reading this register returns the programmed value.

3.13.21 CTI Channel to Trigger 7 Enable Register

The CTIOUTEN7 Register characteristics are:

Purpose Defines which channels can generate a **ctitrigout[7]** output. There is a bit for each of the four channels implemented. This register affects the mapping from application trigger to trigger outputs.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-134 shows the bit assignments.

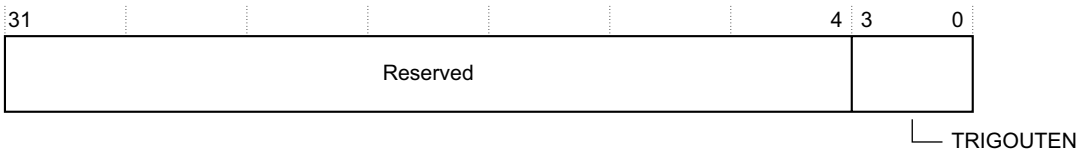


Figure 3-134 CTIOUTEN7 Register bit assignments

Table 3-140 shows the bit assignments.

Table 3-140 CTIOUTEN7 Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	TRIGOUTEN	Changing the value of this bit from a 0 to a 1 enables a channel event for the corresponding channel to generate a **ctitrigout[7]** output. There is one bit of the field for each of the four channels. When a 1 is written to a bit in this register, the channel input, **ctichin**, from the CTM is routed to the **ctitrigout** output. For example, enabling bit 0 enables **ctichin[0]** to cause a trigger event on the **ctitrigout[7]** output. When a 0 is written to any of the bits in this register, the channel input, **ctichin**, from the CTM is not routed to the **ctitrigout** output. Reading this register returns the programmed value.

3.13.22 CTI Trigger In Status Register

The CTITRIGINSTATUS Register characteristics are:

Purpose Provides the status of the **ctitrigin** inputs.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-135 shows the bit assignments.

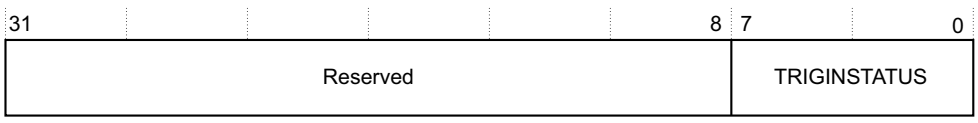


Figure 3-135 CTITRIGINSTATUS Register bit assignments

Table 3-141 shows the bit assignments.

Table 3-141 CTITRIGINSTATUS Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	TRIGINSTATUS	Shows the status of the **ctitrigin** inputs. There is one bit of the field for each trigger input. **1** **ctitrigin** is active. **0** **ctitrigin** is inactive. Because the register provides a view of the raw **ctitrigin** inputs, the reset value is UNKNOWN.

3.13.23 CTI Trigger Out Status Register

The CTITRIGOUTSTATUS Register characteristics are:

Purpose Provides the status of the **ctitrigout** outputs.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-136 shows the bit assignments.

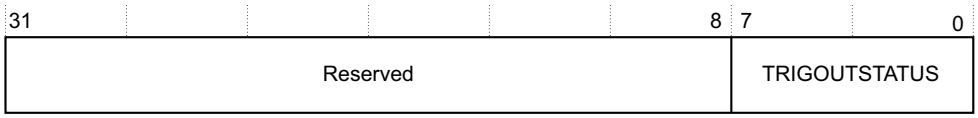


Figure 3-136 CTITRIGOUTSTATUS Register bit assignments

Table 3-142 shows the bit assignments.

Table 3-142 CTITRIGOUTSTATUS Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	TRIGOUTSTATUS	Shows the status of the **ctitrigout** outputs. There is one bit of the field for each trigger output. **1** **ctitrigout** is active. **0** **ctitrigout** is inactive.

3.13.24 CTI Channel In Status Register

The CTICHINSTATUS Register characteristics are:

Purpose Provides the status of the **ctichin** inputs.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-137 on page 3-122 shows the bit assignments.

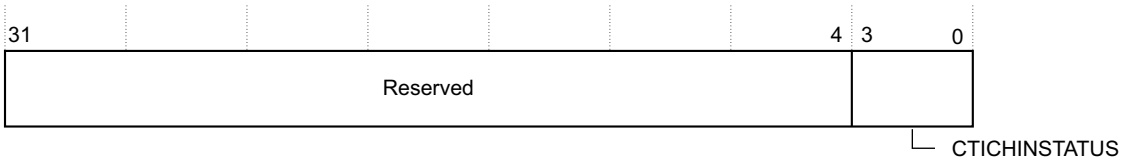


Figure 3-137 CTICHINSTATUS Register bit assignments

Table 3-143 shows the bit assignments.

Table 3-143 CTICHINSTATUS Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	CTICHINSTATUS	Shows the status of the **ctichin** inputs. There is one bit of the field for each channel input. **0** **ctichin** is inactive. **1** **ctichin** is active. Because the register provides a view of the raw **ctichin** inputs, the reset value is UNKNOWN.

3.13.25 CTI Channel Out Status Register

The CTICHOUTSTATUS Register characteristics are:

Purpose Provides the status of the CTI **ctichout** outputs.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-138 shows the bit assignments.

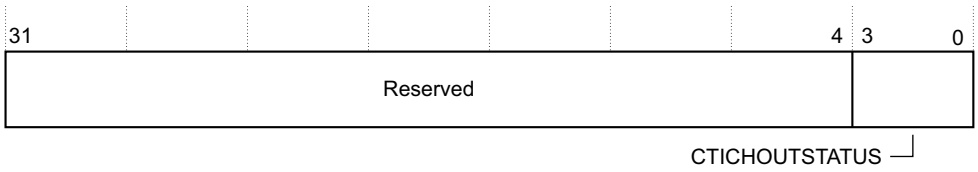


Figure 3-138 CTICHOUTSTATUS Register bit assignments

Table 3-144 shows the bit assignments.

Table 3-144 CTICHOUTSTATUS Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	CTICHOUTSTATUS	Shows the status of the **ctichout** outputs. There is one bit of the field for each channel output. **0** **ctichout** is inactive. **1** **ctichout** is active.

3.13.26 Enable CTI Channel Gate Register

The CTIGATE Register characteristics are:

Purpose Prevents the channels from propagating through the CTM to other CTIs. This enables local cross-triggering, for example for causing an interrupt when the ETM trigger occurs. It can be used effectively with **CTIAPPSET**, **CTIAPPCLEAR**, and **CTIAPPPULSE** for asserting trigger outputs by asserting channels, without affecting the rest of the system. On reset, this register is `0xF`, and channel propagation is enabled.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-139 shows the bit assignments.

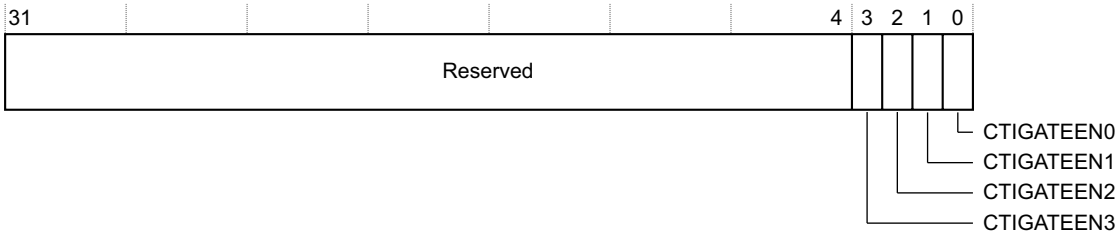


Figure 3-139 CTIGATE Register bit assignments

Table 3-145 shows the bit assignments.

Table 3-145 CTIGATE Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3]	CTIGATEEN3	Enable **ctichout3**. Set to 0 to disable channel propagation.
[2]	CTIGATEEN2	Enable **ctichout2**. Set to 0 to disable channel propagation.
[1]	CTIGATEEN1	Enable **ctichout1**. Set to 0 to disable channel propagation.
[0]	CTIGATEEN0	Enable **ctichout0**. Set to 0 to disable channel propagation.

3.13.27 External Multiplexer Control Register

The ASICCTL Register characteristics are:

Purpose IMPLEMENTATION DEFINED ASIC control. The value written to the register is output on **asicctl[7:0]**.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-140 on page 3-124 shows the bit assignments.

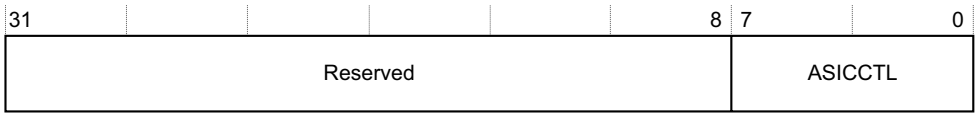


Figure 3-140 ASICCTL Register bit assignments

Table 3-146 shows the bit assignments.

Table 3-146 ASICCTL Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	ASICCTL	If external multiplexing of trigger signals is implemented then the number of multiplexed signals on each trigger must be shown in the Device ID Register. This is done using a Verilog define `EXTMUXNUM`.

3.13.28 Integration Test Channel Input Acknowledge Register

The ITCHINACK Register characteristics are:

Purpose Used to set the value of the **ctchinack** outputs.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-141 shows the bit assignments.

Figure 3-141 ITCHINACK Register bit assignments

Table 3-147 shows the bit assignments.

Table 3-147 ITCHINACK Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	CTCHINACK	Set the value of the **ctchinack** outputs.

3.13.29 Integration Test Trigger Input Acknowledge Register

The ITTRIGINACK Register characteristics are:

Purpose Used to set the value of the **cttriginack** outputs.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-142 shows the bit assignments.

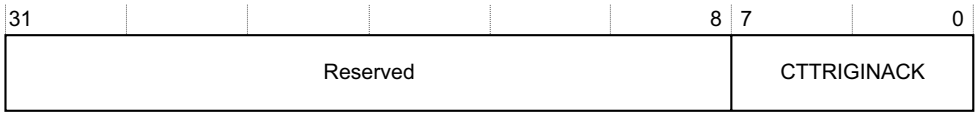


Figure 3-142 ITTRIGINACK Register bit assignments

Table 3-148 shows the bit assignments.

Table 3-148 ITTRIGINACK Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	CTTRIGINACK	Set the value of the **cttriginack** outputs.

3.13.30 Integration Test Channel Output Register

The ITCHOUT Register characteristics are:

Purpose Used to set the value of the **ctchout** outputs.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-143 shows the bit assignments.

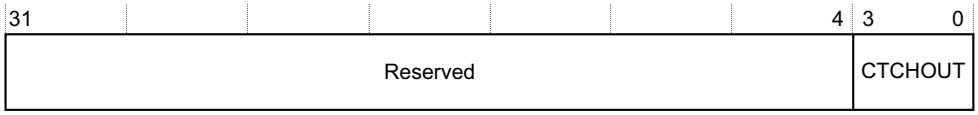


Figure 3-143 ITCHOUT Register bit assignments

Table 3-149 shows the bit assignments.

Table 3-149 ITCHOUT Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	CTCHOUT	Set the value of the **ctchout** outputs.

3.13.31 Integration Test Trigger Output Register

The ITTRIGOUT Register characteristics are:

Purpose Used to set the value of the **cttrigout** outputs.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-144 shows the bit assignments.

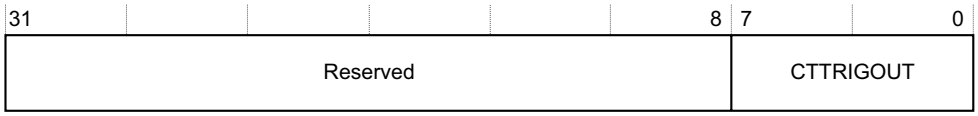


Figure 3-144 ITTRIGOUT Register bit assignments

Table 3-150 shows the bit assignments.

Table 3-150 ITTRIGOUT Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	CTTRIGOUT	Set the value of the **cttrigout** outputs.

3.13.32 Integration Test Channel Output Acknowledge Register

The ITCHOUTACK Register characteristics are:

Purpose Used to read the values of the **ctchoutack** inputs.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-145 shows the bit assignments.

Figure 3-145 ITCHOUTACK Register bit assignments

Table 3-151 shows the bit assignments.

Table 3-151 ITCHOUTACK Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	CTCHOUTACK	Read the values of the **ctchoutack** inputs.

3.13.33 Integration Test Trigger Output Acknowledge Register

The ITTRIGOUTACK Register characteristics are:

Purpose Used to read the values of the **cttrigoutack** inputs.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-146 shows the bit assignments.

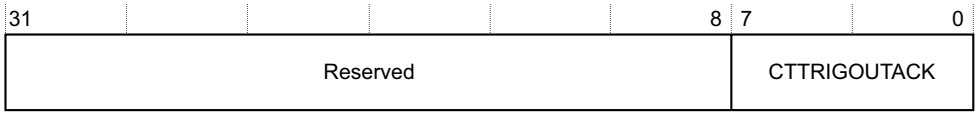


Figure 3-146 ITTRIGOUTACK Register bit assignments

Table 3-152 shows the bit assignments.

Table 3-152 ITTRIGOUTACK Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	CTTRIGOUTACK	Read the value of the **cttrigoutack** inputs.

3.13.34 Integration Test Channel Input Register

The ITCHIN Register characteristics are:

Purpose Used to read the values of the **ctchin** inputs.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-147 shows the bit assignments.

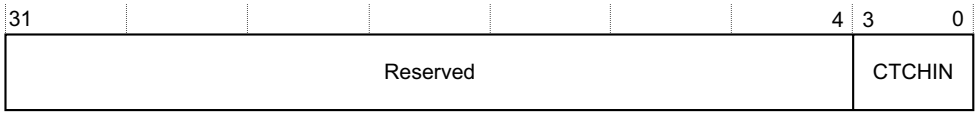


Figure 3-147 ITCHIN Register bit assignments

Table 3-153 shows the bit assignments.

Table 3-153 ITCHIN Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	CTCHIN	Read the value of the **ctchin** inputs.

3.13.35 Integration Test Trigger Input Register

The ITTRIGIN Register characteristics are:

Purpose Used to read the values of the **cttrigin** inputs.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-148 shows the bit assignments.

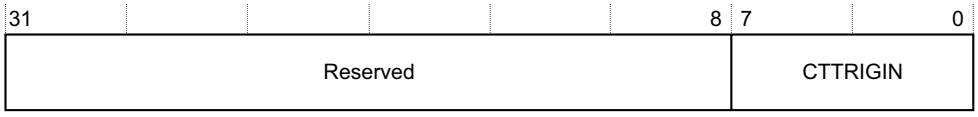


Figure 3-148 ITTRIGIN Register bit assignments

Table 3-154 shows the bit assignments.

Table 3-154 ITTRIGIN Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	CTTRIGIN	Read the values of the **cttrigin** inputs.

3.13.36 Integration Mode Control Register

The ITCTRL Register characteristics are:

Purpose This register enables topology detection. For more information, see the *CoreSight Architecture Specification*. 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 the purposes of integration testing and topology detection.

Note

When a device is in integration mode, the intended functionality might not be available.

After performing integration or topology detection, you must reset the system to ensure correct behavior of CoreSight and other connected system components that the integration or topology detection can affect.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-149 shows the bit assignments.

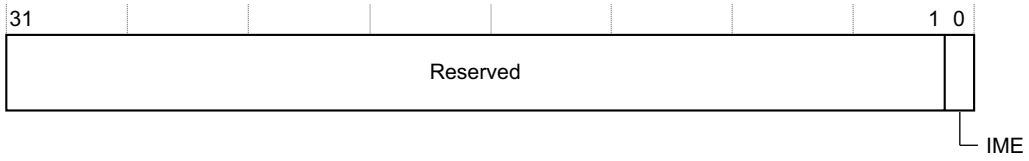


Figure 3-149 ITCTRL Register bit assignments

Table 3-155 shows the bit assignments.

Table 3-155 ITCTRL Register bit assignments

Bits	Name	Function
[31:1]	Reserved	-
[0]	IME	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. **0** Disable integration mode. **1** Enable integration mode.

3.13.37 Claim Tag Set Register

The CLAIMSET Register characteristics are:

Purpose Software can use the claim tag to coordinate application and debugger access to trace unit functionality. The claim tags have no effect on the operation of the component. The CLAIMSET register sets bits in the claim tag, and determines the number of claim bits implemented.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-150 shows the bit assignments.

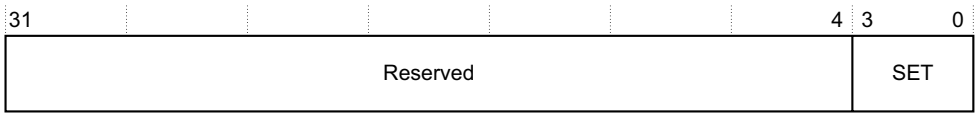


Figure 3-150 CLAIMSET Register bit assignments

Table 3-156 shows the bit assignments.

Table 3-156 CLAIMSET Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	SET_W	When you write 0 to bit[n] of this register, there is no effect on the claim tag bit[n]. When you write 1 to bit[n] of this register, bit[n] in the claim tag is set to 1.
[3:0]	SET_R	When you read 0 on a particular bit of this register, it indicates that this claim tag bit is not implemented. When you read 1 on a particular bit of this register, it indicates that this claim tag bit is implemented. `0b1111` Indicates that four bits of claim tag are implemented.

3.13.38 Claim Tag Clear Register

The CLAIMCLR Register characteristics are:

Purpose Software can use the claim tag to coordinate application and debugger access to trace unit functionality. The claim tags have no effect on the operation of the component. The CLAIMCLR Register clears the bits in the claim tag, and determines the current value of the claim tag.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-151 shows the bit assignments.

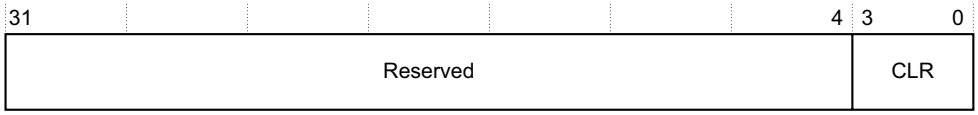


Figure 3-151 CLAIMCLR Register bit assignments

Table 3-157 shows the bit assignments.

Table 3-157 CLAIMCLR Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	CLR_W	When you write 0 to bit[n] of this register, there is no effect on the claim tag bit[n]. When you write 1 to bit[n] of this register, bit[n] in the claim tag is cleared to 0.
[3:0]	CLR_R	The value present reflects the present value of the Claim Tag.

3.13.39 Lock Access Register

The LAR characteristics are:

Purpose Controls write access from self-hosted, on-chip accesses. The LAR does not affect the accesses using the external debugger interface.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-152 shows the bit assignments.

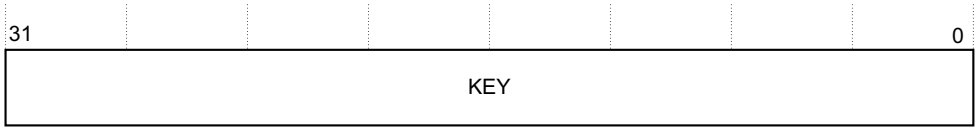


Figure 3-152 LAR bit assignments

Table 3-158 shows the bit assignments.

Table 3-158 LAR bit assignments

Bits	Name	Function
[31:0]	KEY	When you write `0xC5ACCE55`, subsequent write operations to this device are enabled. Any other value disables subsequent write operations.

3.13.40 Lock Status Register

The LSR characteristics are:

Purpose Indicates the status of the lock control mechanism. This lock prevents accidental writes by code being debugged. When locked, write accesses are denied for all registers except for the LAR. The lock registers do not affect accesses from the external debug interface. This register reads as 0 when accessed from the external debug interface.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-153 shows the bit assignments.

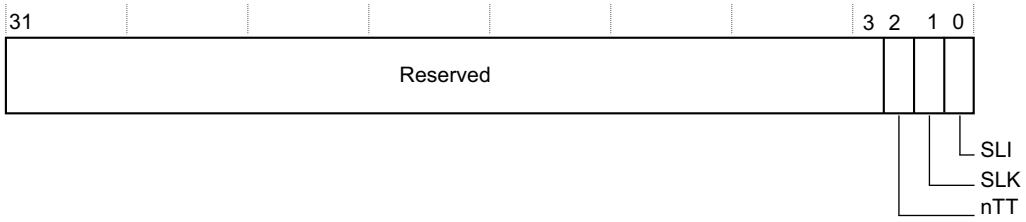


Figure 3-153 LSR bit assignments

Table 3-159 shows the bit assignments.

Table 3-159 LSR bit assignments

Bits	Name	Function
[31:3]	Reserved	-
[2]	nTT	Indicates that the LAR is implemented as 32-bit. `0b0` This component implements a 32-bit LAR Register.
[1]	SLK	Returns the present lock status of the device. `0b0` Indicates that write operations are permitted in this device. `0b1` Indicates that write operations are not permitted in this device. Read operations are permitted.
[0]	SLI	Indicates that a lock control mechanism is present in this device. `0b0` Indicates that a lock control mechanism is not present in this device. Write operations to the LAR are ignored. `0b1` Indicates that a lock control mechanism is present in this device.

3.13.41 Authentication Status Register

The AUTHSTATUS Register characteristics are:

Purpose Reports the required security level and present status.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-154 on page 3-132 shows the bit assignments.

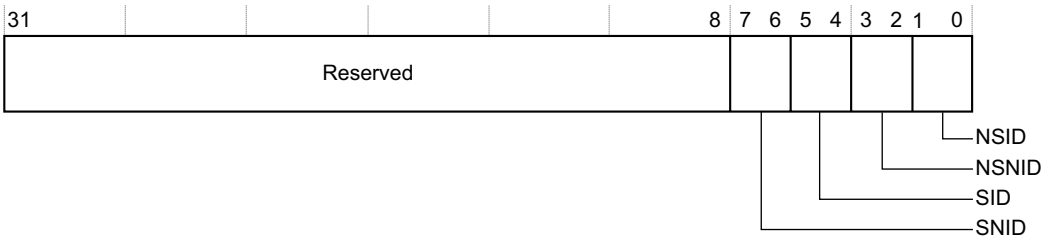


Figure 3-154 AUTHSTATUS Register bit assignments

Table 3-160 shows the bit assignments.

Table 3-160 AUTHSTATUS Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:6]	SNID	Indicates the security level for secure non-invasive debug: 0b00 Functionality is not implemented or is controlled elsewhere.
[5:4]	SID	Indicates the security level for secure invasive debug: 0b00 Functionality is not implemented or is controlled elsewhere.
[3:2]	NSNID	Indicates the security level for non-secure non-invasive debug: 0b10 Functionality is disabled. 0b11 Functionality is enabled.
[1:0]	NSID	Indicates the security level for non-secure invasive debug: 0b10 Functionality is disabled. 0b11 Functionality is enabled.

3.13.42 Device Configuration Register

The DEVID Register characteristics are:

Purpose Indicates the capabilities of the component.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-155 shows the bit assignments.

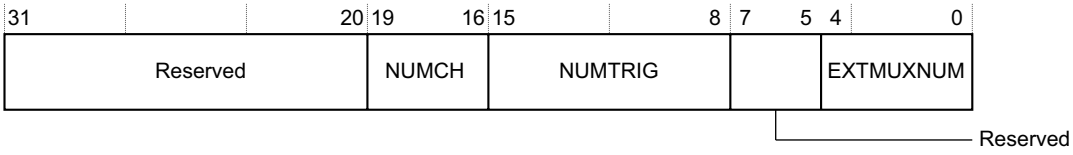


Figure 3-155 DEVID Register bit assignments

Table 3-161 shows the bit assignments.

Table 3-161 DEVID Register bit assignments

Bits	Name	Function
[31:20]	Reserved	-
[19:16]	NUMCH	Number of ECT channels available.
[15:8]	NUMTRIG	Number of ECT triggers available.
[7:5]	Reserved	-
[4:0]	EXTMUXNUM	Indicates the number of multiplexing available on Trigger Inputs and Trigger Outputs using **asicctl**. The default value of `0b00000` indicates that no multiplexing is present. Reflects the value of the Verilog define `EXTMUXNUM` that you must alter accordingly.

3.13.43 Device Type Identifier Register

The DEVTYPE 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 See the register summary in Table 3-119 on page 3-104.

Figure 3-156 shows the bit assignments.

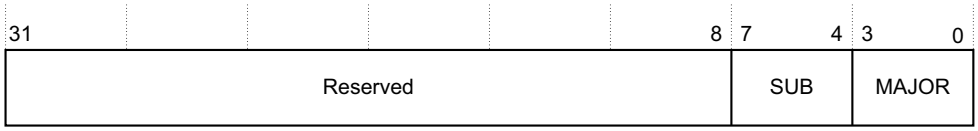


Figure 3-156 DEVTYPE Register bit assignments

Table 3-162 shows the bit assignments.

Table 3-162 DEVTYPE Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	SUB	Sub-classification of the type of the debug component as specified in the *CoreSight Architecture Specification* within the major classification as specified in the MAJOR field. `0b0001` Indicates that this component is a cross-triggering component.
[3:0]	MAJOR	Major classification of the type of the debug component as specified in the *CoreSight Architecture Specification* for this debug and trace component. `0b0100` Indicates that this component allows a debugger to control other components in a CoreSight system.

3.13.44 Peripheral ID4 Register

The PIDR4 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-157 shows the bit assignments.

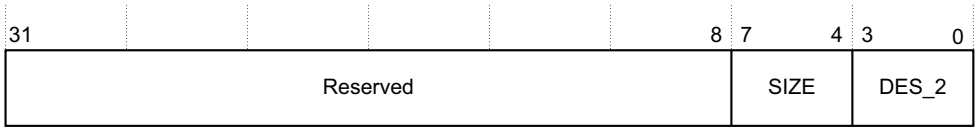


Figure 3-157 PIDR4 bit assignments

Table 3-163 shows the bit assignments.

Table 3-163 PIDR4 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	SIZE	This is a 4-bit value that indicates the total contiguous size of the memory window this component uses in powers of 2 from the standard 4KB. `0b0000` Indicates that the device only occupies 4KB of memory.
[3:0]	DES_2	This is the JEDEC JEP106 continuation code. This code, along with bits[6:4] of the identity code defined in the PIDR0 Register and bits[3:0] of the identity code defined in the PIDR1 Register, identifies the designer of the component. `0b0100` JEDEC continuation code.

3.13.45 Peripheral ID5-7 Registers

The PIDR5-7 characteristics are:

Purpose Reserved.

Usage constraints There are no usage constraints.

Configurations These registers are available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-158 shows the bit assignments.

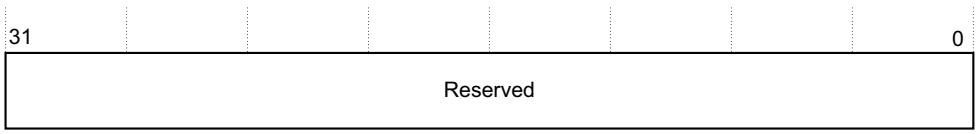


Figure 3-158 PIDR5-7 bit assignments

Table 3-164 shows the bit assignments.

Table 3-164 PIDR5-7 bit assignments

Bits	Name	Function
[31:0]	Reserved	-

3.13.46 Peripheral ID0 Register

The PIDR0 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 See the register summary in Table 3-119 on page 3-104.

Figure 3-159 shows the bit assignments.

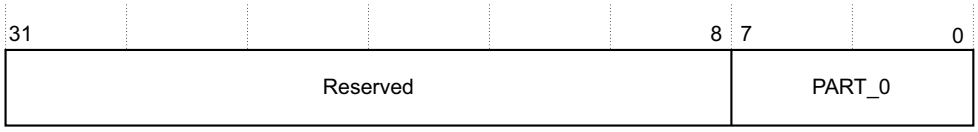


Figure 3-159 PIDR0 bit assignments

Table 3-165 shows the bit assignments.

Table 3-165 PIDR0 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PART_0	Bits[7:0] of the 12-bit part number of the component. The designer of the component assigns this part number. `0x06` Indicates bits[7:0] of the part number of the component.

3.13.47 Peripheral ID1 Register

The PIDR1 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 See the register summary in Table 3-119 on page 3-104.

Figure 3-160 on page 3-136 shows the bit assignments.

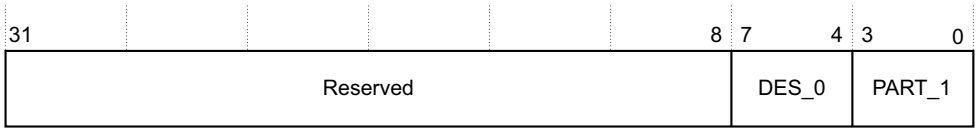


Figure 3-160 PIDR1 bit assignments

Table 3-166 shows the bit assignments.

Table 3-166 PIDR1 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	DES_0	Bits[3:0] of the JEDEC JEP106 identity code. This code, along with bits[6:4] of the identity code defined in the PIDR2 and the continuation code defined in the PIDR4, identifies the designer of the component. `0b1011` Bits[3:0] of the JEDEC JEP106 Identity Code. The default value is ARM.
[3:0]	PART_1	Bits[11:8] of the 12-bit part number of the component. The designer of the component assigns this part number. `0b1001` Indicates bits[11:8] of the part number of the component.

3.13.48 Peripheral ID2 Register

The PIDR2 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 See the register summary in Table 3-119 on page 3-104.

Figure 3-161 shows the bit assignments.

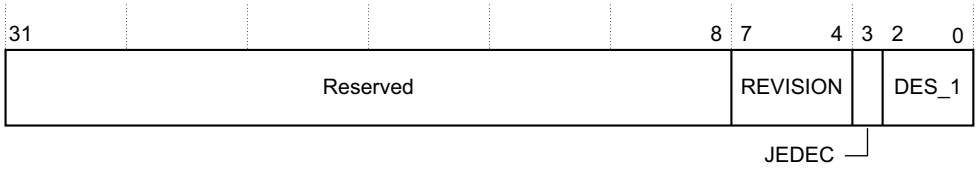


Figure 3-161 PIDR2 bit assignments

Table 3-167 shows the bit assignments.

Table 3-167 PIDR2 bit assignments

Bits	Name	Function
[31:8]	Reserved	-

Table 3-167 PIDR2 bit assignments (continued)

Bits	Name	Function
[7:4]	REVISION	An incremental value starting from 0b0000 for the first revision of this component. This increases by 1 for both major and minor revisions and is used to identify the major or minor revisions. 0b0100 This device is at r0p4.
[3]	JEDEC	Always set. Indicates whether the JEDEC assigned designer ID is used. 0b1 Indicates that a JEDEC assigned designer ID is used.
[2:0]	DES_1	Bits[6:4] of the JEDEC JEP106 identity code. This code, along with bits[3:0] of the identity code defined in the PIDR1 and the continuation code defined in the PIDR4, identifies the designer of the component. 0b011 Bits[6:4] of the JEDEC JEP106 Identity Code. The default value is ARM.

3.13.49 Peripheral ID3 Register

The PIDR3 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-162 shows the bit assignments.

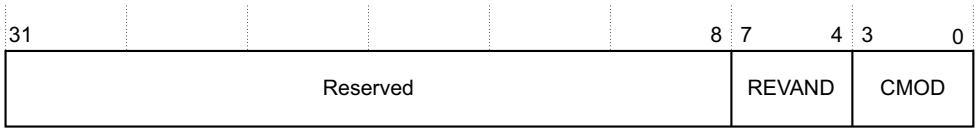


Figure 3-162 PIDR3 bit assignments

Table 3-168 shows the bit assignments.

Table 3-168 PIDR3 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	REVAND	Indicates minor errata fixes specific to the revision of the component being used, for example metal fixes after implementation. In most cases, this field is zero. ARM recommends that the component designers ensure that a metal fix can change this field if required, for example, by driving it from registers that reset to zero. 0b0000 Indicates that there are no errata fixes to this component.
[3:0]	CMOD	Indicates whether the customer has modified the behavior of the component. In most cases, this field is zero. The customer changes this value on modifications to this component. 0b0000 Indicates that the customer has not modified this component.

3.13.50 Component ID0 Register

The CIDR0 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-163 shows the bit assignments.

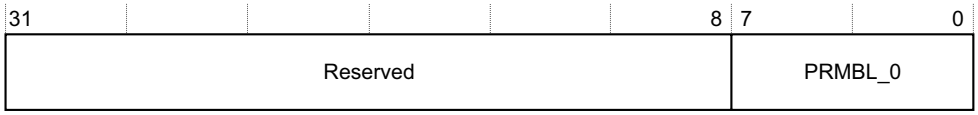


Figure 3-163 CIDR0 bit assignments

Table 3-169 shows the it assignments.

Table 3-169 CIDR0 bit assignments

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

3.13.51 Component ID1 Register

The CIDR1 characteristics are:

Purpose A component identification register that indicates 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 See the register summary in Table 3-119 on page 3-104.

Figure 3-164 shows the bit assignments.

Figure 3-164 CIDR1 bit assignments

Table 3-170 shows the bit assignments.

Table 3-170 CIDR1 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	CLASS	Class of the component, for example, if the component is a ROM table or a generic CoreSight component. Contains bits[15:12] of the component identification code. 0b1001 Indicates that the component is a CoreSight component.
[3:0]	PRMBL_1	Contains bits[11:8] of the component identification code. 0b0000 Bits[11:8] of the identification code.

3.13.52 Component ID2 Register

The CIDR2 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-165 shows the bit assignments.

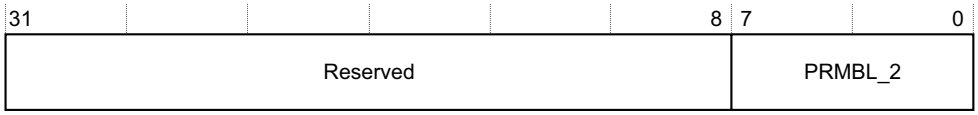


Figure 3-165 CIDR2 bit assignments

Table 3-171 shows the bit assignments.

Table 3-171 CIDR2 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_2	Contains bits[23:16] of the component identification code. 0x05 Bits[23:16] of the identification code.

3.13.53 Component ID3 Register

The CIDR3 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-119 on page 3-104.

Figure 3-166 shows the bit assignments.

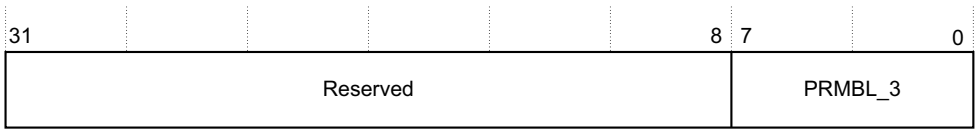


Figure 3-166 CIDR3 bit assignments

Table 3-172 shows the bit assignments.

Table 3-172 CIDR3 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_3	Contains bits[31:24] of the component identification code. `0xB1` Bits[31:24] of the identification code.

3.14 TPIU register summary

Table 3-173 shows the TPIU registers in offset order from the base memory address.

Table 3-173 TPIU register summary

Offset	Name	Type	Reset	Description
0x000	Supported_Port_Sizes	RO	0x00000001	*Supported Port Size Register* on page 3-143
0x004	Current_port_size	RW	0x00000001	*Current Port Size Register* on page 3-147
0x100	Supported_trigger_modes	RO	0x0000011F	*Supported Trigger Modes Register* on page 3-151
0x104	Trigger_counter_value	RW	0x00000000	*Trigger Counter Value Register* on page 3-152
0x108	Trigger_multiplier	RW	0x00000000	*Trigger Multiplier Register* on page 3-153
0x200	Supported_test_pattern_modes	RO	0x0003000F	*Supported Test Patterns/Modes Register* on page 3-154
0x204	Current_test_pattern_mode	RW	0x00000000	*Current Test Pattern/Modes Register* on page 3-155
0x208	TPRCR	RW	0x00000000	*TPIU Test Pattern Repeat Counter Register* on page 3-156
0x300	FFSR	RO	0x00000000	*Formatter and Flush Status Register* on page 3-157
0x304	FFCR	RW	0x00000000	*Formatter and Flush Control Register* on page 3-158
0x308	FSCR	RW	0x00000040	*Formatter Synchronization Counter Register* on page 3-160
0x400	EXTCTL_In_Port	RO	0x00000000	*TPIU EXCTL Port Register - In* on page 3-161
0x404	EXTCTL_Out_Port	RW	0x00000000	*TPIU EXCTL Port Register - Out* on page 3-162
0xEE4	ITTRFLINACK	WO	0x00000000	*Integration Test Trigger In and Flush In Acknowledge Register* on page 3-162
0xEE8	ITTRFLIN	RO	0x00000000	*Integration Test Trigger In and Flush In Register* on page 3-163
0xEEC	ITATBDATA0	RO	0x00000000	*Integration Test ATB Data Register 0* on page 3-164
0xEF0	ITATBCTR2	WO	0x00000000	*Integration Test ATB Control Register 2* on page 3-165
0xEF4	ITATBCTR1	RO	0x00000000	*Integration Test ATB Control Register 1* on page 3-166
0xEF8	ITATBCTR0	RO	0x00000000	*Integration Test ATB Control Register 0* on page 3-166
0xF00	ITCTRL	RW	0x00000000	*Integration Mode Control Register* on page 3-167
0xFA0	CLAIMSET	RW	0x0000000F	*Claim Tag Set Register* on page 3-168
0xFA4	CLAIMCLR	RW	0x00000000	*Claim Tag Clear Register* on page 3-168
0xFB0	LAR	WO	0x00000000	*Lock Access Register* on page 3-169
0xFB4	LSR	RO	0x00000003	*Lock Status Register* on page 3-170
0xFB8	AUTHSTATUS	RO	0x00000000	*Authentication Status Register* on page 3-170
0xFC8	DEVID	RO	0x000000A0	*Device Configuration Register* on page 3-171
0xFCC	DEVTYPE	RO	0x00000011	*Device Type Identifier Register* on page 3-172
0xFD0	PIDR4	RO	0x00000004	*Peripheral ID4 Register* on page 3-173

Table 3-173 TPIU register summary (continued)

Offset	Name	Type	Reset	Description
0xFD4	PIDR5	RO	0x00000000	*Peripheral ID5-7 registers* on page 3-173
0xFD8	PIDR6	RO	0x00000000	
0xFDC	PIDR7	RO	0x00000000	
0xFE0	PIDR0	RO	0x00000012	*Peripheral ID0 Register* on page 3-174
0xFE4	PIDR1	RO	0x000000B9	*Peripheral ID1 Register* on page 3-175
0xFE8	PIDR2	RO	0x0000004B	*Peripheral ID2 Register* on page 3-175
0xFEC	PIDR3	RO	0x00000000	*Peripheral ID3 Register* on page 3-176
0xFF0	CIDR0	RO	0x0000000D	*Component ID0 Register* on page 3-177
0xFF4	CIDR1	RO	0x00000090	*Component ID1 Register* on page 3-177
0xFF8	CIDR2	RO	0x00000005	*Component ID2 Register* on page 3-178
0xFFC	CIDR3	RO	0x000000B1	*Component ID3 Register* on page 3-178

3.15 TPIU register descriptions

This section describes the TPIU registers. Table 3-173 on page 3-141 provides cross references to individual registers.

3.15.1 Supported Port Size Register

The Supported_Port_Sizes Register characteristics are:

Purpose Each bit location represents a single port size that is supported on the device, that is, 32-1 in bit locations [31:0]. If the bit is set then that port size is permitted. By default the RTL is designed to support all port sizes, set to `0xFFFFFFFF`.

This register reflects the value of the `CSTPIU_SUPPORTSIZE_VAL` Verilog define value, currently not user-modifiable, and is more constrained by the input tie-off **tpmaxdatasize**. The external tie-off, **tpmaxdatasize**, must be set during finalization of the ASIC to reflect the actual number of **tracedata** signals being wired to physical pins. This is to ensure that tools do not attempt to select a port width that cannot be captured by an attached *Trace Port Analyzer* (TPA).

The value on **tpmaxdatasize** causes bits within the Supported Port Size register that represent wider widths to be clear, that is, unsupported.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-167 on page 3-144 shows the bit assignments.

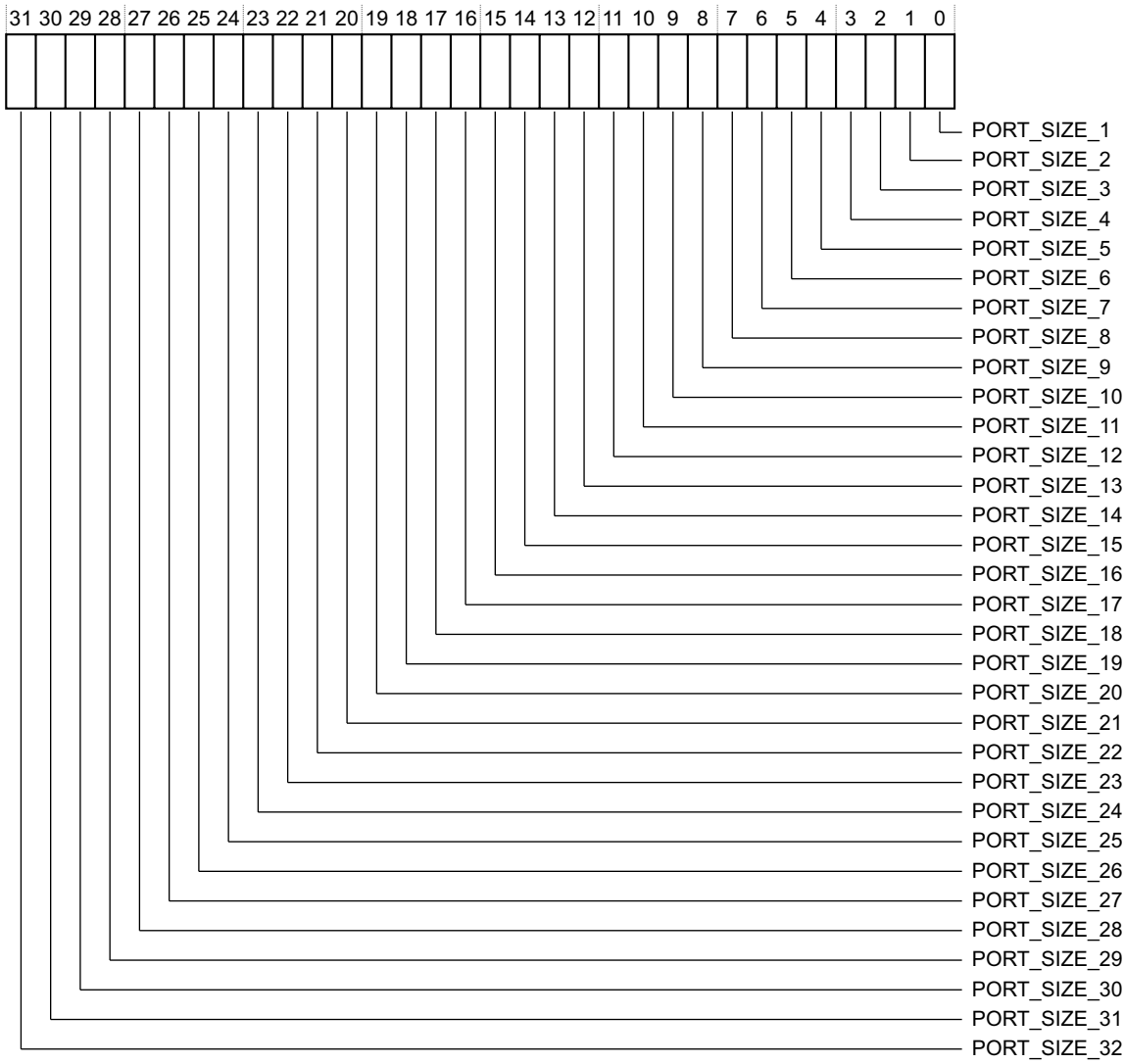


Figure 3-167 Supported_Port_Sizes Register bit assignments

Table 3-174 shows the bit assignments.

Table 3-174 Supported_Port_Sizes Register bit assignments

Bits	Name	Function
[31]	PORT_SIZE_32	Indicates whether the TPIU supports port size of 32-bit. **0** Port size is not supported. **1** Port size is supported.
[30]	PORT_SIZE_31	Indicates whether the TPIU supports port size of 31-bit. **0** Port size is not supported. **1** Port size is supported.
[29]	PORT_SIZE_30	Indicates whether the TPIU supports port size of 30-bit. **0** Port size is not supported. **1** Port size is supported.

Table 3-174 Supported_Port_Sizes Register bit assignments (continued)

Bits	Name	Function
[28]	PORT_SIZE_29	Indicates whether the TPIU supports port size of 29-bit. **0** Port size is not supported. **1** Port size is supported.
[27]	PORT_SIZE_28	Indicates whether the TPIU supports port size of 28-bit. **0** Port size is not supported. **1** Port size is supported.
[26]	PORT_SIZE_27	Indicates whether the TPIU supports port size of 27-bit. **0** Port size is not supported. **1** Port size is supported.
[25]	PORT_SIZE_26	Indicates whether the TPIU supports port size of 26-bit. **0** Port size is not supported. **1** Port size is supported.
[24]	PORT_SIZE_25	Indicates whether the TPIU supports port size of 25-bit. **0** Port size is not supported. **1** Port size is supported.
[23]	PORT_SIZE_24	Indicates whether the TPIU supports port size of 24-bit. **0** Port size is not supported. **1** Port size is supported.
[22]	PORT_SIZE_23	Indicates whether the TPIU supports port size of 23-bit. **0** Port size is not supported. **1** Port size is supported.
[21]	PORT_SIZE_22	Indicates whether the TPIU supports port size of 22-bit. **0** Port size is not supported. **1** Port size is supported.
[20]	PORT_SIZE_21	Indicates whether the TPIU supports port size of 21-bit. **0** Port size is not supported. **1** Port size is supported.
[19]	PORT_SIZE_20	Indicates whether the TPIU supports port size of 20-bit. **0** Port size is not supported. **1** Port size is supported.
[18]	PORT_SIZE_19	Indicates whether the TPIU supports port size of 19-bit. **0** Port size is not supported. **1** Port size is supported.
[17]	PORT_SIZE_18	Indicates whether the TPIU supports port size of 18-bit. **0** Port size is not supported. **1** Port size is supported.
[16]	PORT_SIZE_17	Indicates whether the TPIU supports port size of 17-bit. **0** Port size is not supported. **1** Port size is supported.

Table 3-174 Supported_Port_Sizes Register bit assignments (continued)

Bits	Name	Function
[15]	PORT_SIZE_16	Indicates whether the TPIU supports port size of 16-bit. **0** Port size is not supported. **1** Port size is supported.
[14]	PORT_SIZE_15	Indicates whether the TPIU supports port size of 15-bit. **0** Port size is not supported. **1** Port size is supported.
[13]	PORT_SIZE_14	Indicates whether the TPIU supports port size of 14-bit. **0** Port size is not supported. **1** Port size is supported.
[12]	PORT_SIZE_13	Indicates whether the TPIU supports port size of 13-bit. **0** Port size is not supported. **1** Port size is supported.
[11]	PORT_SIZE_12	Indicates whether the TPIU supports port size of 12-bit. **0** Port size is not supported. **1** Port size is supported.
[10]	PORT_SIZE_11	Indicates whether the TPIU supports port size of 11-bit. **0** Port size is not supported. **1** Port size is supported.
[9]	PORT_SIZE_10	Indicates whether the TPIU supports port size of 10-bit. **0** Port size is not supported. **1** Port size is supported.
[8]	PORT_SIZE_9	Indicates whether the TPIU supports port size of 9-bit. **0** Port size is not supported. **1** Port size is supported.
[7]	PORT_SIZE_8	Indicates whether the TPIU supports port size of 8-bit. **0** Port size is not supported. **1** Port size is supported.
[6]	PORT_SIZE_7	Indicates whether the TPIU supports port size of 7-bit. **0** Port size is not supported. **1** Port size is supported.
[5]	PORT_SIZE_6	Indicates whether the TPIU supports port size of 6-bit. **0** Port size is not supported. **1** Port size is supported.
[4]	PORT_SIZE_5	Indicates whether the TPIU supports port size of 5-bit. **0** Port size is not supported. **1** Port size is supported.
[3]	PORT_SIZE_4	Indicates whether the TPIU supports port size of 4-bit. **0** Port size is not supported. **1** Port size is supported.

Table 3-174 Supported_Port_Sizes Register bit assignments (continued)

Bits	Name	Function
[2]	PORT_SIZE_3	Indicates whether the TPIU supports port size of 3-bit. **0** Port size is not supported. **1** Port size is supported.
[1]	PORT_SIZE_2	Indicates whether the TPIU supports port size of 2-bit. **0** Port size is not supported. **1** Port size is supported.
[0]	PORT_SIZE_1	Indicates whether the TPIU supports port size of 1-bit. **0** Port size is not supported. **1** Port size is supported.

3.15.2 Current Port Size Register

The Current_port_size Register characteristics are:

Purpose Has the same format as the Supported Port Sizes register but only one bit is set, and all others must be zero. Writing values with more than one bit set or setting a bit that is not indicated as supported is not supported and causes UNPREDICTABLE behavior. On reset this defaults to the smallest possible port size, 1 bit, and so reads as `0x00000001`.

Note

Do not modify the value while the Trace Port is still active, or without correctly stopping the formatter. See *Formatter and Flush Control Register* on page 3-158. This can result in data not being aligned to the port width. For example, data on an 8-bit Trace Port might not be byte aligned.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-168 on page 3-148 shows the bit assignments.

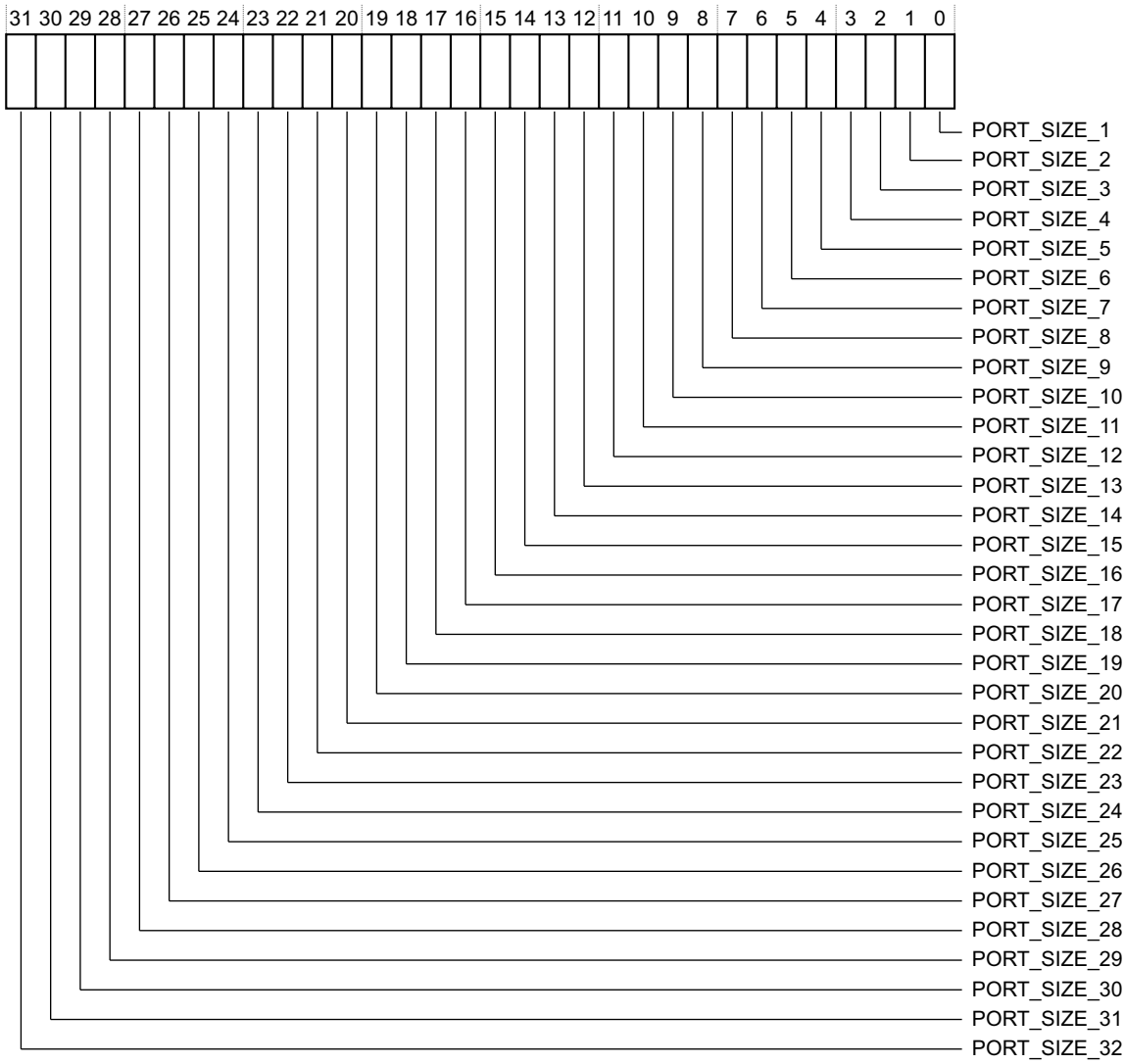


Figure 3-168 Current_port_size Register bit assignments

Table 3-175 shows the bit assignments.

Table 3-175 Current_port_size Register bit assignments

Bits	Name	Function
[31]	PORT_SIZE_32	Indicates whether the current port size of the TPIU is 32-bit. **0** Current port size is not 32. **1** Current port size is 32.
[30]	PORT_SIZE_31	Indicates whether the current port size of the TPIU is 31-bit. **0** Current port size is not 31. **1** Current port size is 31.
[29]	PORT_SIZE_30	Indicates whether the current port size of the TPIU is 30-bit. **0** Current port size is not 30. **1** Current port size is 30.

Table 3-175 Current_port_size Register bit assignments (continued)

Bits	Name	Function
[28]	PORT_SIZE_29	Indicates whether the current port size of the TPIU is 29-bit. **0** Current port size is not 29. **1** Current port size is 29.
[27]	PORT_SIZE_28	Indicates whether the current port size of the TPIU is 28-bit. **0** Current port size is not 28. **1** Current port size is 28.
[26]	PORT_SIZE_27	Indicates whether the current port size of the TPIU is 27-bit. **0** Current port size is not 27. **1** Current port size is 27.
[25]	PORT_SIZE_26	Indicates whether the current port size of the TPIU is 26-bit. **0** Current port size is not 26. **1** Current port size is 26.
[24]	PORT_SIZE_25	Indicates whether the current port size of the TPIU is 25-bit. **0** Current port size is not 25. **1** Current port size is 25.
[23]	PORT_SIZE_24	Indicates whether the current port size of the TPIU is 24-bit. **0** Current port size is not 24. **1** Current port size is 24.
[22]	PORT_SIZE_23	Indicates whether the current port size of the TPIU is 23-bit. **0** Current port size is not 23. **1** Current port size is 23.
[21]	PORT_SIZE_22	Indicates whether the current port size of the TPIU is 22-bit. **0** Current port size is not 22. **1** Current port size is 22.
[20]	PORT_SIZE_21	Indicates whether the current port size of the TPIU is 21-bit. **0** Current port size is not 21. **1** Current port size is 21.
[19]	PORT_SIZE_20	Indicates whether the current port size of the TPIU is 20-bit. **0** Current port size is not 20. **1** Current port size is 20.
[18]	PORT_SIZE_19	Indicates whether the current port size of the TPIU is 19-bit. **0** Current port size is not 19. **1** Current port size is 19.
[17]	PORT_SIZE_18	Indicates whether the current port size of the TPIU is 18-bit. **0** Current port size is not 18. **1** Current port size is 18.
[16]	PORT_SIZE_17	Indicates whether the current port size of the TPIU is 17-bit. **0** Current port size is not 17. **1** Current port size is 17.

Table 3-175 Current_port_size Register bit assignments (continued)

Bits	Name	Function
[15]	PORT_SIZE_16	Indicates whether the current port size of the TPIU is 16-bit. **0** Current port size is not 16. **1** Current port size is 16.
[14]	PORT_SIZE_15	Indicates whether the current port size of the TPIU is 15-bit. **0** Current port size is not 15. **1** Current port size is 15.
[13]	PORT_SIZE_14	Indicates whether the current port size of the TPIU is 14-bit. **0** Current port size is not 14. **1** Current port size is 14.
[12]	PORT_SIZE_13	Indicates whether the current port size of the TPIU is 13-bit. **0** Current port size is not 13. **1** Current port size is 13.
[11]	PORT_SIZE_12	Indicates whether the current port size of the TPIU is 12-bit. **0** Current port size is not 12. **1** Current port size is 12.
[10]	PORT_SIZE_11	Indicates whether the current port size of the TPIU is 11-bit. **0** Current port size is not 11. **1** Current port size is 11.
[9]	PORT_SIZE_10	Indicates whether the current port size of the TPIU is 10-bit. **0** Current port size is not 10. **1** Current port size is 10.
[8]	PORT_SIZE_9	Indicates whether the current port size of the TPIU is 9-bit. **0** Current Port size is not 9. **1** Current Port size is 9.
[7]	PORT_SIZE_8	Indicates whether the current port size of the TPIU is 8-bit. **0** Current port size is not 8. **1** Current port size is 8.
[6]	PORT_SIZE_7	Indicates whether the current port size of the TPIU is 7-bit. **0** Current port size is not 7. **1** Current port size is 7.
[5]	PORT_SIZE_6	Indicates whether the current port size of the TPIU is 6-bit. **0** Current port size is not 6. **1** Current port size is 6.
[4]	PORT_SIZE_5	Indicates whether the current port size of the TPIU is 5-bit. **0** Current port size is not 5. **1** Current port size is 5.
[3]	PORT_SIZE_4	Indicates whether the current port size of the TPIU is 4-bit. **0** Current port size is not 4. **1** Current port size is 4.

Table 3-175 Current_port_size Register bit assignments (continued)

Bits	Name	Function
[2]	PORT_SIZE_3	Indicates whether the current port size of the TPIU is 3-bit. **0** Current port size is not 3. **1** Current port size is 3.
[1]	PORT_SIZE_2	Indicates whether the current port size of the TPIU is 2-bit. **0** Current port size is not 2. **1** Current port size is 2.
[0]	PORT_SIZE_1	Indicates whether the current port size of the TPIU is 1-bit. **0** Current port size is not 1. **1** Current port size is 1.

3.15.3 Supported Trigger Modes Register

The Supported_trigger_modes Register characteristics are:

Purpose Indicates the implemented trigger counter multipliers and other supported features of the trigger system.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-169 shows the bit assignments.

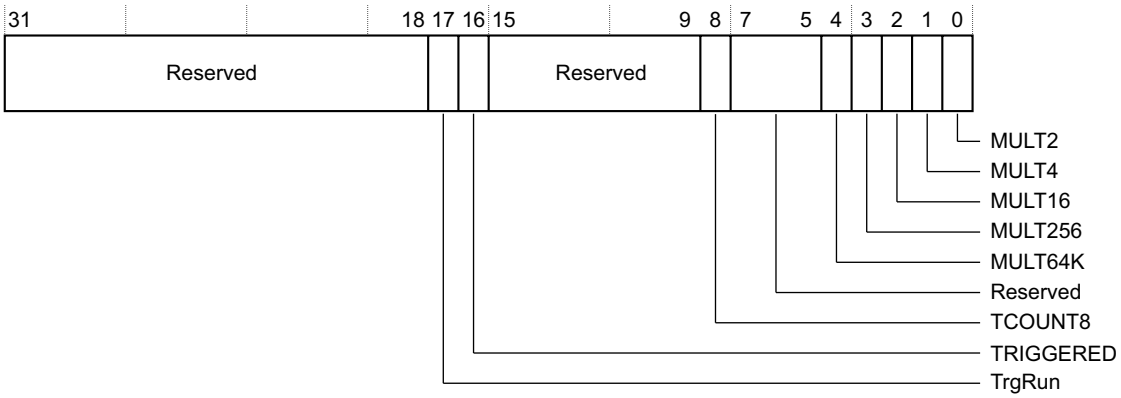


Figure 3-169 Supported_trigger_modes Register bit assignments

Table 3-176 shows the bit assignments.

Table 3-176 Supported_trigger_modes Register bit assignments

Bits	Name	Function
[31:18]	Reserved	-
[17]	TrgRun	A trigger has occurred but the counter is not at zero. **0** Either a trigger has not occurred or the counter is at zero. **1** A trigger has occurred but the counter is not at zero.
[16]	TRIGGERED	A trigger has occurred and the counter has reached zero. **0** Trigger has not occurred. **1** Trigger has occurred.
[15:9]	Reserved	-
[8]	TCOUNT8	Indicates whether an 8-bit wide counter register is implemented. **0** 8-bit wide counter register is not implemented. **1** 8-bit wide counter register is implemented.
[7:5]	Reserved	-
[4]	MULT64K	Indicates whether multiply the trigger counter by 65536 is supported. **0** Multiply the trigger counter by 65536 not supported. **1** Multiply the trigger counter by 65536 supported.
[3]	MULT256	Indicates whether multiply the trigger counter by 256 is supported. **0** Multiply the trigger counter by 256 not supported. **1** Multiply the trigger counter by 256 supported.
[2]	MULT16	Indicates whether multiply the trigger counter by 16 is supported. **0** Multiply the trigger counter by 16 not supported. **1** Multiply the trigger counter by 16 supported.
[1]	MULT4	Indicates whether multiply the trigger counter by 4 is supported. **0** Multiply the trigger counter by 4 not supported. **1** Multiply the trigger counter by 4 supported.
[0]	MULT2	Indicates whether multiply the trigger counter by 2 is supported. **0** Multiply the trigger counter by 2 not supported. **1** Multiply the trigger counter by 2 supported.

3.15.4 Trigger Counter Value Register

The Trigger_counter_value Register characteristics are:

Purpose Enables delaying the indication of triggers to any external connected trace capture or storage devices. This counter is only eight bits wide and is intended to be used only with the counter multipliers in the Trigger Multiplier Register, `0x108`. When a trigger is started, this value, in combination with the multiplier, is the number of words before the trigger is indicated. When the trigger counter reaches zero, the value written here is reloaded. Writing to this register causes the trigger counter value to reset but not reset any values on the multiplier. Reading this register returns the preset value, not the current count.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-170 shows the bit assignments.

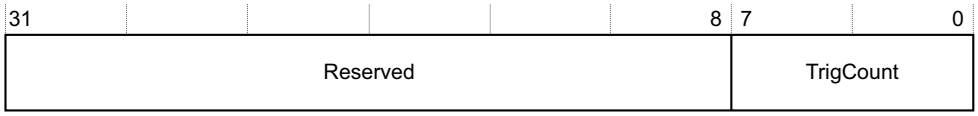


Figure 3-170 Trigger_counter_value Register bit assignments

Table 3-177 shows the bit assignments.

Table 3-177 Trigger_counter_value Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	TrigCount	8-bit counter value for the number of words to be output from the formatter before a trigger is inserted. At reset the value is zero and this value has the effect of disabling the register, that is, there is no delay.

3.15.5 Trigger Multiplier Register

The Trigger_multiplier Register characteristics are:

Purpose Contains the selectors for the trigger counter multiplier. Several multipliers can be selected to create the required multiplier value, that is, any value between one and approximately 2x10^9. The default value is multiplied by one, `0x0`. Writing to this register causes the internal trigger counter and the state in the multipliers to be reset to initial count position, that is, the trigger counter is reloaded with the Trigger Counter Register value and all multipliers are reset.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-171 shows the Trigger_multiplier Register bit assignments.

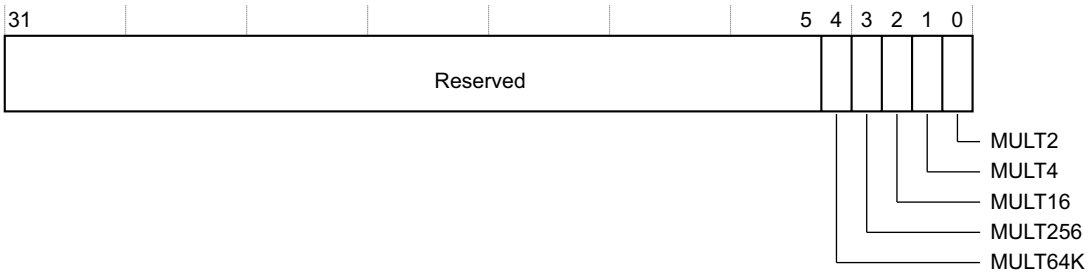


Figure 3-171 Trigger_multiplier Register bit assignments

Table 3-178 shows the Trigger_multiplier Register bit assignments.

Table 3-178 Trigger_multiplier Register bit assignments

Bits	Name	Function
[31:5]	Reserved	-
[4]	MULT64K	Multiply the Trigger Counter by 65536 (2^16). **0** Multiplier disabled. **1** Multiplier enabled.
[3]	MULT256	Multiply the Trigger Counter by 256 (2^8). **0** Multiplier disabled. **1** Multiplier enabled.
[2]	MULT16	Multiply the Trigger Counter by 16 (2^4). **0** Multiplier disabled. **1** Multiplier enabled.
[1]	MULT4	Multiply the Trigger Counter by 4 (2^2). **0** Multiplier disabled. **1** Multiplier enabled.
[0]	MULT2	Multiply the Trigger Counter by 2 (2^1). **0** Multiplier disabled. **1** Multiplier enabled.

3.15.6 Supported Test Patterns/Modes Register

The Supported_test_pattern_modes Register characteristics are:

Purpose Provides a set of known bit sequences or patterns that can be output over the trace port and be detected by the TPA or other associated trace capture device.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-172 shows the bit assignments.

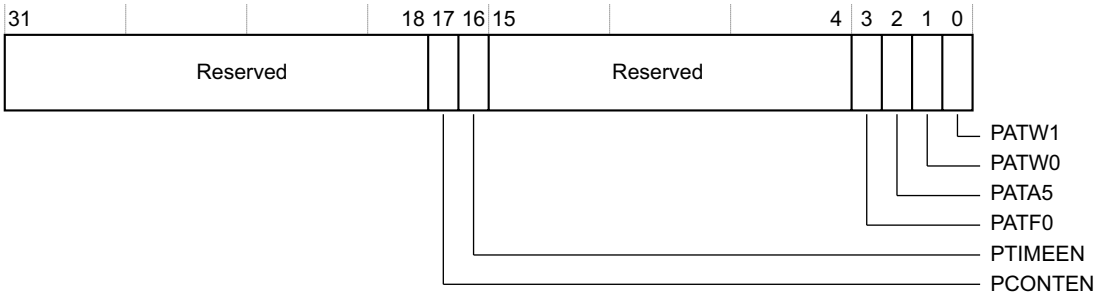


Figure 3-172 Supported_test_pattern_modes Register bit assignments

Table 3-179 shows the bit assignments.

Table 3-179 Supported_test_pattern_modes Register bit assignments

Bits	Name	Function
[31:18]	Reserved	-
[17]	PCONTEN	Indicates whether continuous mode is supported. **0** Mode not supported. **1** Mode supported.
[16]	PTIMEEN	Indicates whether timed mode is supported. **0** Mode not supported. **1** Mode supported.
[15:4]	Reserved	-
[3]	PATF0	FF/00 pattern supported to be output over the trace port. **0** Pattern not supported. **1** Pattern supported.
[2]	PATA5	AA/55 pattern supported to be output over the trace port. **0** Pattern not supported. **1** Pattern supported.
[1]	PATW0	Walking 0s Pattern supported to be output over the trace port. **0** Pattern not supported. **1** Pattern supported.
[0]	PATW1	Walking 1s Pattern supported to be output over the trace port. **0** Pattern not supported. **1** Pattern supported.

3.15.7 Current Test Pattern/Modes Register

The Current_test_pattern_mode Register characteristics are:

Purpose Indicates the current test pattern or mode selected. Only one of the modes can be set, using bits[17:16], but a multiple number of bits for the patterns can be set using bits[3:0]. If timed mode is selected, after the allotted number of cycles is reached, the mode automatically switches to off mode. On reset, this register is set to `18'h00000`, off mode with no selected patterns.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-173 on page 3-156 shows the bit assignments.

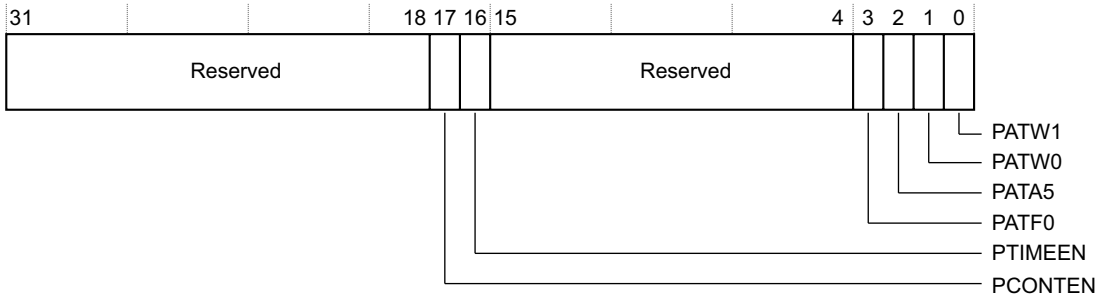


Figure 3-173 Current_test_pattern_mode Register bit assignments

Table 3-180 shows the bit assignments.

Table 3-180 Current_test_pattern_mode Register bit assignments

Bits	Name	Function
[31:18]	Reserved	-
[17]	PCONTEN	Indicates whether Continuous Mode is enabled. **0** Mode disabled. **1** Mode enabled.
[16]	PTIMEEN	Indicates whether Timed Mode is enabled. **0** Mode disabled. **1** Mode enabled.
[15:4]	Reserved	-
[3]	PATF0	FF/00 pattern enabled to be output over the Trace Port. **0** Pattern disabled. **1** Pattern enabled.
[2]	PATA5	AA/55 pattern enabled to be output over the Trace Port. **0** Pattern disabled. **1** Pattern enabled.
[1]	PATW0	Walking 0s Pattern enabled to be output over the Trace Port. **0** Pattern disabled. **1** Pattern enabled.
[0]	PATW1	Walking 1s Pattern enabled to be output over the Trace Port. **0** Pattern disabled. **1** Pattern enabled.

3.15.8 TPIU Test Pattern Repeat Counter Register

The TPRCR Register characteristics are:

Purpose An 8-bit counter start value that is decremented. A write sets the initial counter value and a read returns the programmed value. On reset this value is set to 0.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-174 shows the bit assignments.

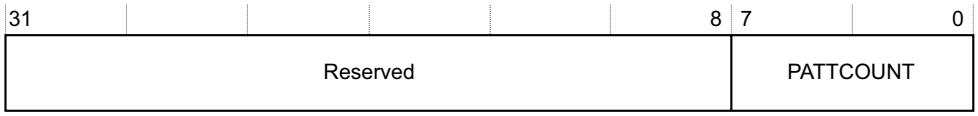


Figure 3-174 TPRCR Register bit assignments

Table 3-181 shows the bit assignments.

Table 3-181 TPRCR Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PATTCOUNT	8-bit counter value to indicate the number of **traceclkin** cycles that a pattern runs for before switching to the next pattern. The default value is 0.

3.15.9 Formatter and Flush Status Register

The FFSR Register characteristics are:

Purpose Indicates the current status of the formatter and flush features available in the TPIU.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-175 shows the bit assignments.

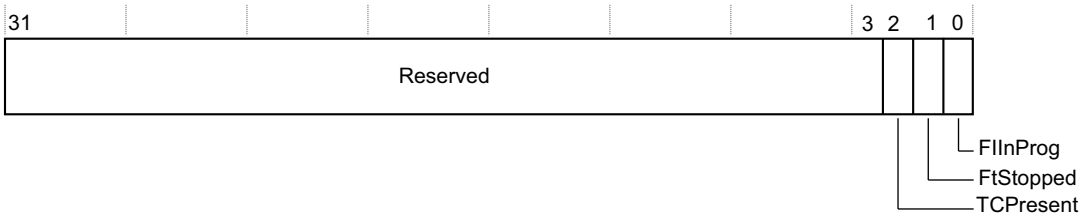


Figure 3-175 FFSR Register bit assignments

Table 3-182 shows the FFSR Register bit assignments.

Table 3-182 FFSR Register bit assignments

Bits	Name	Function
[31:3]	Reserved	-
[2]	TCPresent	Indicates whether the **tracectl** pin is available for use.If this bit is set then **tracectl** is present. If no **tracectl** pin is available, that is, this bit is zero, then the data formatter must be used and only in continuous mode. If either constraint reports LOW, this bit shows that no **tracectl** is present and this inability to use the pin is reflected in this register. **0** **tracectl** pin not present. **1** **tracectl** pin present.
[1]	FtStopped	The formatter has received a stop request signal and all trace data and post-amble is sent. Any more trace data on the ATB interface is ignored and **atreadys** goes HIGH. **0** Formatter has not stopped. **1** Formatter has stopped.
[0]	FlInProg	This is an indication of the current state of **afvalids**. **0** **afvalids** is LOW. **1** **afvalids** is HIGH.

3.15.10 Formatter and Flush Control Register

The FFCR Register characteristics are:

Purpose Controls the generation of stop, trigger, and flush events. To disable formatting and put the formatter into bypass mode, bits 1 and 0 must be clear. Setting both bits is the same as setting bit 1. All three flush generating conditions can be enabled together. However, if a second or third flush event is generated from another condition then the current flush completes before the next flush is serviced.

Flush from **flushin** takes priority over flush from trigger, which in turn completes before a manually activated flush. All Trigger indication conditions can be enabled simultaneously although this can cause the appearance of multiple triggers if flush using trigger is also enabled. Both `Stop On` settings can be enabled although if flush on trigger is set up, none of the flushed data is stored. When the system stops, it returns **atreadys** and does not store the accepted data packets. This is to avoid stalling of any other devices that are connected to a trace replicator. If an event in the Formatter and Flush Control Register is required, it must be enabled before the originating event starts. Because requests from flushes and triggers can originate in an asynchronous clock domain, the exact time the component acts on the request cannot be determined with respect to configuring the control. To perform a stop on flush completion through a manually generated flush request, two write operations to the register are required:

- One to enable the stop event, if it is not already enabled
- One to generate the manual flush.

Note

ARM recommends that you change the trace port width without enabling continuous mode. Enabling continuous mode causes data to be sent from the trace port and modifying the port size can result in data not being aligned for power 2 port widths.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-176 shows the bit assignments.

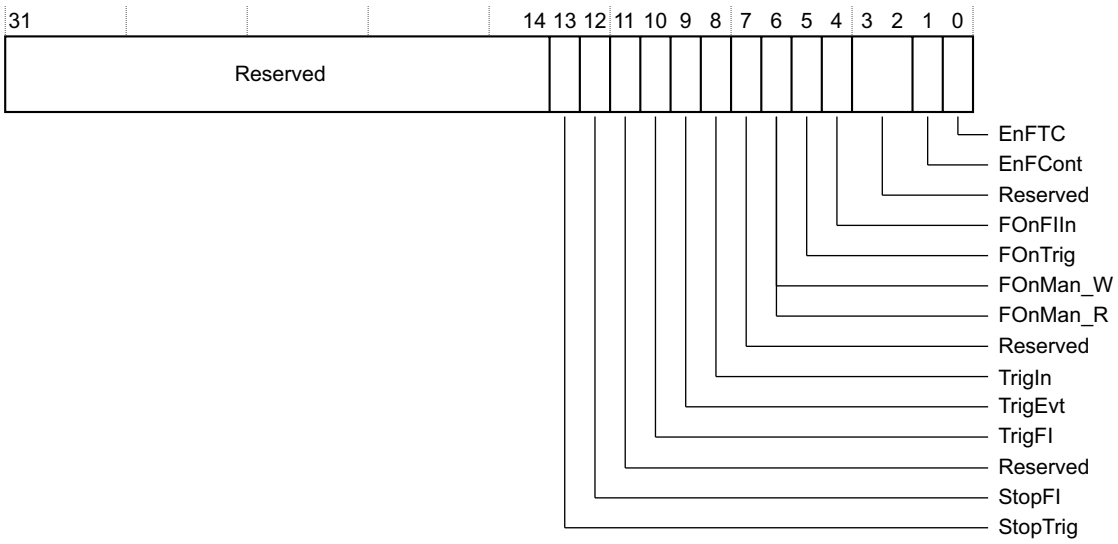


Figure 3-176 FFCR Register bit assignments

Table 3-183 shows the bit assignments.

Table 3-183 FFCR Register bit assignments

Bits	Name	Function
[31:14]	Reserved	-
[13]	StopTrig	Stops the formatter after a trigger event is observed. Reset to disabled or zero. **0** Disable stopping the formatter after a trigger event is observed. **1** Enable stopping the formatter after a trigger event is observed.
[12]	StopFl	Forces the FIFO to drain off any part-completed packets. Setting this bit enables this function but it is cleared on reset, or disabled. **0** Disable stopping the formatter on return of **afreadys**. **1** Enable stopping the formatter on return of **afreadys**.
[11]	Reserved	-
[10]	TrigFl	Indicates a trigger when flush completion on **afreadys** is returned. **0** Disable trigger indication on return of **afreadys**. **1** Enable trigger indication on return of **afreadys**.

Table 3-183 FFCR Register bit assignments (continued)

Bits	Name	Function
[9]	TrigEvt	Indicate a trigger on a trigger event. **0** Disable trigger indication on a trigger event. **1** Enable trigger indication on a trigger event.
[8]	TrigIn	Indicate a trigger when **trigin** is asserted. **0** Disable trigger indication when **trigin** is asserted. **1** Enable trigger indication when **trigin** is asserted.
[7]	Reserved	-
[6]	FOnMan_W	Setting this bit generates a flush. This is cleared when this flush is serviced. This bit is cleared on reset. **0** Manual flush is not initiated. **1** Manual flush is initiated.
[6]	FOnMan_R	Setting this bit generates a flush. This is cleared when this flush is serviced. This bit is cleared on reset. **0** Manual flush is not initiated. **1** Manual flush is initiated.
[5]	FOnTrig	Setting this bit initiates a manual flush of data in the system when a trigger event occurs. On reset this bit is clear. A trigger event is defined as when the trigger counter reaches zero, or in the case of the trigger counter being zero, when **trigin** is HIGH. **0** Disable generation of flush when a Trigger Event occurs. **1** Enable generation of flush when a Trigger Event occurs.
[4]	FOnFlIn	Setting this bit enables the use of the **flushin** connection. This bit is cleared on reset. **0** Disable generation of flush using the **flushin** interface. **1** Enable generation of flush using the **flushin** interface.
[3:2]	Reserved	-
[1]	EnFCont	Embed in trigger packets and indicate null cycles using sync packets. This bit is cleared on reset. This setting can only be changed when **FtStopped** is HIGH. **0** Continuous formatting disabled. **1** Continuous formatting enabled.
[0]	EnFTC	Do not embed triggers into the formatted stream. Trace disable cycles and triggers are indicated by **tracectl**, where present. This bit is cleared on reset. This setting can only be changed when **FtStopped** is HIGH. **0** Formatting disabled. **1** Formatting enabled.

3.15.11 Formatter Synchronization Counter Register

The FSCR Register characteristics are:

Purpose Enables effective use on different sized TPAs without wasting large amounts of the storage capacity of the capture device. This counter is the number of formatter frames since the last synchronization packet of 128 bits, and is a 12-bit counter with a maximum count value of 4096. This equates to synchronization every 65536 bytes, that is, 4096 packets x 16 bytes per packet. The default is set up for a synchronization packet every 1024 bytes, that is, every 64 formatter frames. If the formatter is configured for continuous mode, full and half-word sync frames are

inserted during normal operation. Under these circumstances, the count value represents the maximum number of complete frames between full synchronization packets.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-177 shows the bit assignments.

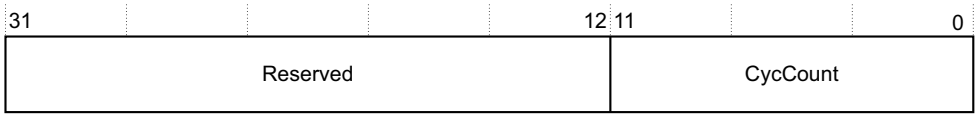


Figure 3-177 FSCR Register bit assignments

Table 3-184 shows the bit assignments.

Table 3-184 FSCR Register bit assignments

Bits	Name	Function
[31:12]	Reserved	-
[11:0]	CycCount	12-bit counter value that indicates the number of complete frames between full synchronization packets. The default value is `0x40`.

3.15.12 TPIU EXCTL Port Register - In

The EXTCTL_In_Port Register characteristics are:

Purpose Two ports can be used as a control and feedback mechanism for any serializers, pin sharing multiplexers or, other solutions that might be added to the trace output pins either for pin control or a high speed trace port solution. These ports are raw register banks that sample or export the corresponding external pins. The output register bank is set to all zeros on reset. The input registers sample the incoming signals and are therefore UNDEFINED.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-178 shows the bit assignments.

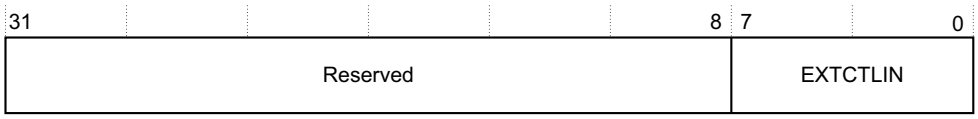


Figure 3-178 EXTCTL_In_Port Register bit assignments

Table 3-185 shows the bit assignments.

Table 3-185 EXTCTL_In_Port Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	EXTCTLIN	EXTCTL inputs.

3.15.13 TPIU EXCTL Port Register - Out

The EXTCTL_Out_Port Register characteristics are:

Purpose Two ports can be used as a control and feedback mechanism for any serializers, pin sharing multiplexers, or other solutions that might be added to the trace output pins either for pin control or a high speed trace port solution. These ports are raw register banks that sample or export the corresponding external pins. The output register bank is set to all zeros on reset. The input registers sample the incoming signals and are therefore UNDEFINED.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-179 shows the bit assignments.

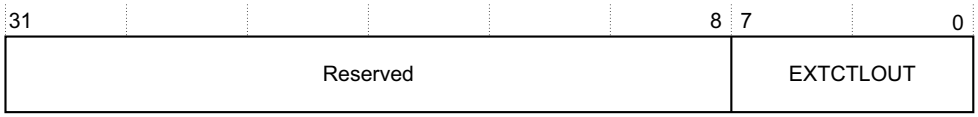


Figure 3-179 EXTCTL_Out_Port Register bit assignments

Table 3-186 shows the bit assignments.

Table 3-186 EXTCTL_Out_Port Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	EXTCTLOUT	EXTCTL outputs.

3.15.14 Integration Test Trigger In and Flush In Acknowledge Register

The ITTRFLINACK Register characteristics are:

Purpose Enables control of the **triginack** and **flushinack** outputs from the TPIU.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-180 on page 3-163 shows the bit assignments.

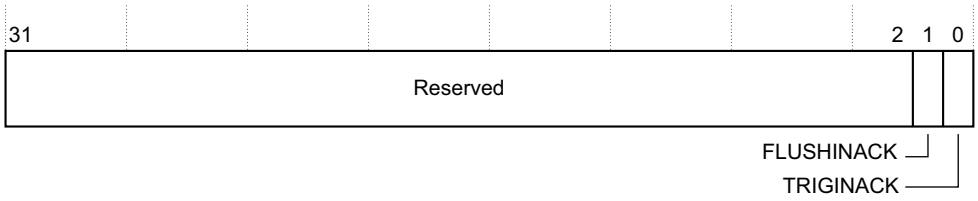


Figure 3-180 ITTRFLINACK Register bit assignments

Table 3-187 shows the bit assignments.

Table 3-187 ITTRFLINACK Register bit assignments

Bits	Name	Function
[31:2]	Reserved	-
[1]	FLUSHINACK	Sets the value of **flushinack**. **0** Set the value of **flushinack** to 0. **1** Set the value of **flushinack** to 1.
[0]	TRIGINACK	Sets the value of **triginack**. **0** Set the value of **triginack** to 0. **1** Set the value of **triginack** to 1.

3.15.15 Integration Test Trigger In and Flush In Register

The ITTRFLIN Register characteristics are:

Purpose Contains the values of the **flushin** and **trigin** inputs to the TPIU.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-181 shows the bit assignments.

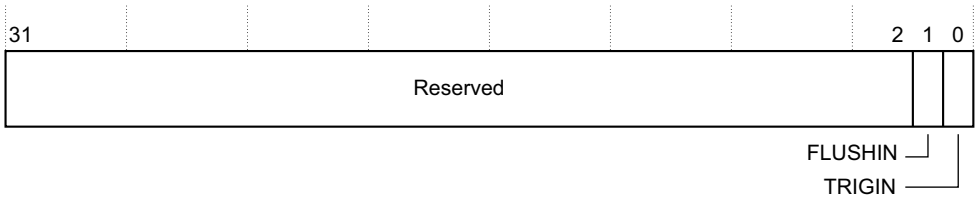


Figure 3-181 ITTRFLIN Register bit assignments

Table 3-188 shows the bit assignments.

Table 3-188 ITTRFLIN Register bit assignments

Bits	Name	Function
[31:2]	Reserved	-
[1]	FLUSHIN	Reads the value of **flushin**. **0** **flushin** is LOW. **1** **flushin** is HIGH.
[0]	TRIGIN	Reads the value of **trigin**. **0** **trigin** is LOW. **1** **trigin** is HIGH.

3.15.16 Integration Test ATB Data Register 0

The ITATBDATA0 Register characteristics are:

Purpose Contains the value of the **atdatas** inputs to the TPIU. The values are valid only when **atvalids** is HIGH.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-182 shows the bit assignments.

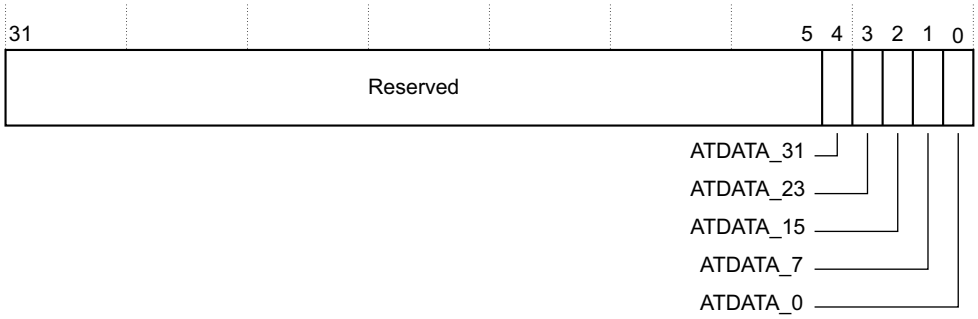


Figure 3-182 ITATBDATA0 Register bit assignments

Table 3-189 shows the bit assignments.

Table 3-189 ITATBDATA0 Register bit assignments

Bits	Name	Function
[31:5]	Reserved	-
[4]	ATDATA_31	Reads the value of **atdatas[31]**. **1** **atdatas[31]** is 1. **0** **atdatas[31]** is 0.
[3]	ATDATA_23	Read the value of **atdatas[23]**. **1** **atdatas[23]** is 1. **0** **atdatas[23]** is 0.

Table 3-189 ITATBDATA0 Register bit assignments (continued)

Bits	Name	Function
[2]	ATDATA_15	Reads the value of **atdatas[15]**. **1** **atdatas[15]** is 1. **0** **atdatas[15]** is 0.
[1]	ATDATA_7	Reads the value of **atdatas[7]**. **1** **atdatas[7]** is 1. **0** **atdatas[7]** is 0.
[0]	ATDATA_0	Reads the value of **atdatas[0]**. **1** **atdatas[0]** is 1. **0** **atdatas[0]** is 0.

3.15.17 Integration Test ATB Control Register 2

The ITATBCTR2 Register characteristics are:

Purpose Enables control of the **atreadys** and **afvalids** outputs of the TPIU.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-183 shows the bit assignments.

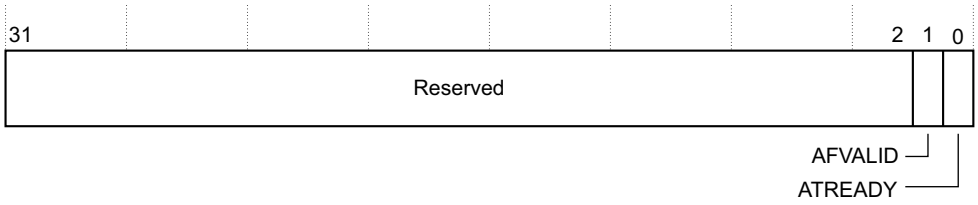


Figure 3-183 ITATBCTR2 Register bit assignments

Table 3-190 shows the bit assignments.

Table 3-190 ITATBCTR2 Register bit assignments

Bits	Name	Function
[31:2]	Reserved	-
[1]	AFVALID	Sets the value of **afvalid**. **0** Set the value of **afvalid** to 0. **1** Set the value of **afvalid** to 1.
[0]	ATREADY	Sets the value of **atready**. **0** Set the value of **atready** to 0. **1** Set the value of **atready** to 1.

3.15.18 Integration Test ATB Control Register 1

The ITATBCTR1 Register characteristics are:

Purpose Contains the value of the **atids** input to the TPIU. This is only valid when **atvalids** is HIGH.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-184 shows the bit assignments.

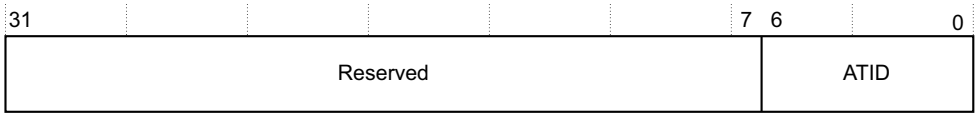


Figure 3-184 ITATBCTR1 Register bit assignments

Table 3-191 shows the bit assignments.

Table 3-191 ITATBCTR1 Register bit assignments

Bits	Name	Function
[31:7]	Reserved	-
[6:0]	ATID	Reads the value of **atids**.

3.15.19 Integration Test ATB Control Register 0

The ITATBCTR0 Register characteristics are:

Purpose Captures the values of the **atvalids**, **afreadys**, and **atbytess** inputs to the TPIU. To ensure the integration registers work correctly in a system, the value of **atbytess** is only valid when **atvalids**, bit[0], is HIGH.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-185 shows the bit assignments.

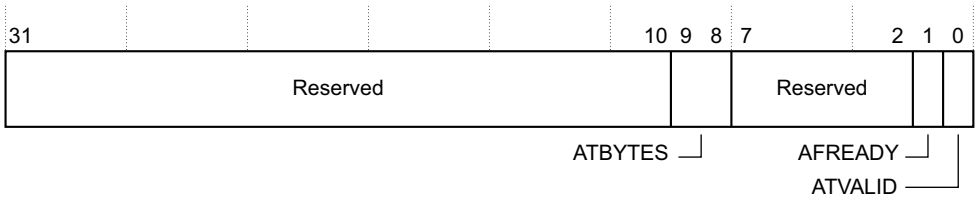


Figure 3-185 ITATBCTR0 Register bit assignments

Table 3-192 shows the bit assignments.

Table 3-192 ITATBCTR0 Register bit assignments

Bits	Name	Function
[31:10]	Reserved	-
[9:8]	ATBYTES	Reads the value of **atbytess**.
[7:2]	Reserved	-
[1]	AFREADY	Reads the value of **afreadys**. **0** **afreadys** is 0. **1** **afreadys** is 1.
[0]	ATVALID	Reads the value of **atvalids**. **0** **atvalids** is 0. **1** **atvalids** is 1.

3.15.20 Integration Mode Control Register

The ITCTRL Register characteristics are:

Purpose Enables topology detection. For more information, see the *CoreSight Architecture Specification*. 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 detection.

Note

When a device is in integration mode, the intended functionality might not be available.

After performing integration or topology detection, you must reset the system to ensure correct behavior of CoreSight and other connected system components that the integration or topology detection can affect.

The registers in the TPIU enable the system to set the **flushinack** and **triginack** output pins. The **flushin** and **trigin** inputs to the TPIU can also be read. The other Integration Test Registers are for testing the integration of the ATB slave interface on the TPIU.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-186 shows the bit assignments.

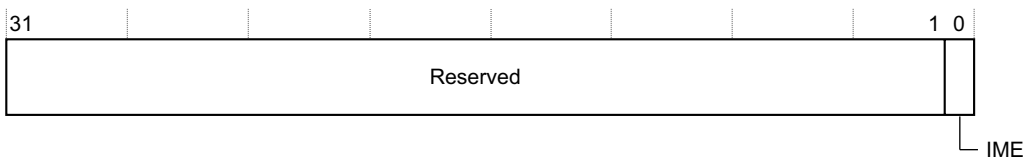


Figure 3-186 ITCTRL Register bit assignments

Table 3-193 shows the bit assignments.

Table 3-193 ITCTRL Register bit assignments

Bits	Name	Function
[31:1]	Reserved	-
[0]	IME	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. **0** Disable integration mode. **1** Enable integration mode.

3.15.21 Claim Tag Set Register

The CLAIMSET Register characteristics are:

Purpose Software can use the claim tag to coordinate application and debugger access to trace unit functionality. The claim tags have no effect on the operation of the component. The CLAIMSET Register sets bits in the claim tag, and determines the number of claim bits implemented.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-187 shows the bit assignments.

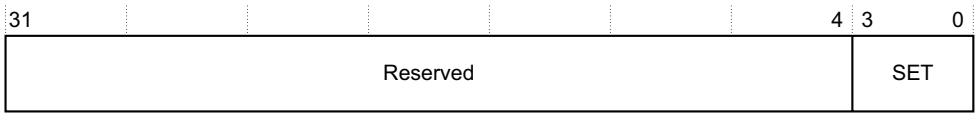


Figure 3-187 CLAIMSET Register bit assignments

Table 3-194 shows the bit assignments.

Table 3-194 CLAIMSET Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	SET_W	When you write 0 to bit[n] of this register, there is no effect on the claim tag bit[n]. When you write 1 to bit[n] of this register, bit[n] in the claim tag is set to 1.
[3:0]	SET_R	When you read 0 on a particular bit of this register, it indicates that this claim tag bit is not implemented. When you read 1 on a particular bit of this register, it indicates that this claim tag bit is implemented. `0b1111` Indicates that four bits of claim tag are implemented.

3.15.22 Claim Tag Clear Register

The CLAIMCLR Register characteristics are:

Purpose Software can use the claim tag to coordinate application and debugger access to trace unit functionality. The claim tags have no effect on the operation of the component. The CLAIMCLR Register clears the bits in the claim tag, and determines the current value of the claim tag.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-188 shows the bit assignments.

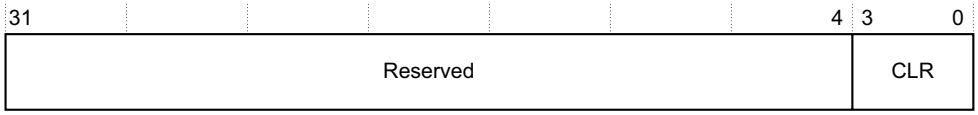


Figure 3-188 CLAIMCLR Register bit assignments

Table 3-195 shows the bit assignments.

Table 3-195 CLAIMCLR Register bit assignments

Bits	Name	Function
[31:4]	Reserved	-
[3:0]	CLR_W	When you write 0 to bit[n] of this register, there is no effect on the claim tag bit[n]. When you write 1 to bit[n] of this register, bit[n] in the claim tag is cleared to 0.
[3:0]	CLR_R	The value present reflects the present value of the Claim tag.

3.15.23 Lock Access Register

The LAR characteristics are:

Purpose Controls write access from self-hosted, on-chip accesses. The LAR does not affect the accesses using the external debugger interface.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-189 shows the bit assignments.

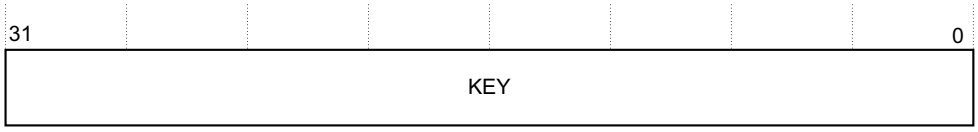


Figure 3-189 LAR Register bit assignments

Table 3-196 shows the bit assignments.

Table 3-196 LAR bit assignments

Bits	Name	Function
[31:0]	KEY	When you write `0xC5ACCE55`, subsequent write operations to this device are enabled. Any other value disables subsequent write operations.

3.15.24 Lock Status Register

The LSR characteristics are:

Purpose Indicates the status of the lock control mechanism. This lock prevents accidental writes by code being debugged. When locked, write accesses are denied for all registers except for the LAR. The lock registers do not affect accesses from the external debug interface. This register reads as 0 when accessed from the external debug interface.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-190 shows the bit assignments.

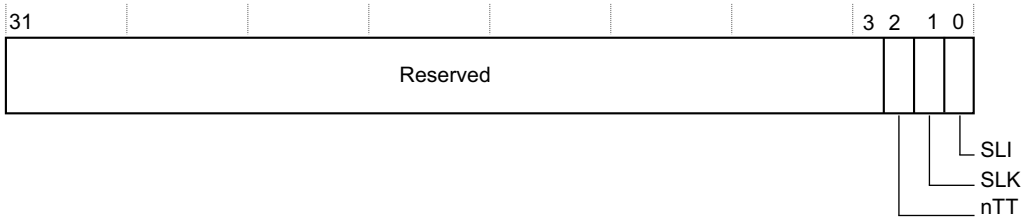


Figure 3-190 LSR bit assignments

Table 3-197 shows the bit assignments.

Table 3-197 LSR bit assignments

Bits	Name	Function
[31:3]	Reserved	-
[2]	nTT	Indicates that the LAR is implemented as 32-bit. `0b0` This component implements a 32-bit LAR.
[1]	SLK	Returns the present lock status of the device. `0b0` Indicates that write operations are permitted in this device. `0b1` Indicates that write operations are not permitted in this device. Read operations are permitted.
[0]	SLI	Indicates that a lock control mechanism is present in this device. `0b0` Indicates that a lock control mechanism is not present in this device. Write operations to the LAR Register are ignored. `0b1` Indicates that a lock control mechanism is present in this device.

3.15.25 Authentication Status Register

The AUTHSTATUS Register characteristics are:

Purpose Reports the required security level and present status.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-191 on page 3-171 shows the bit assignments.

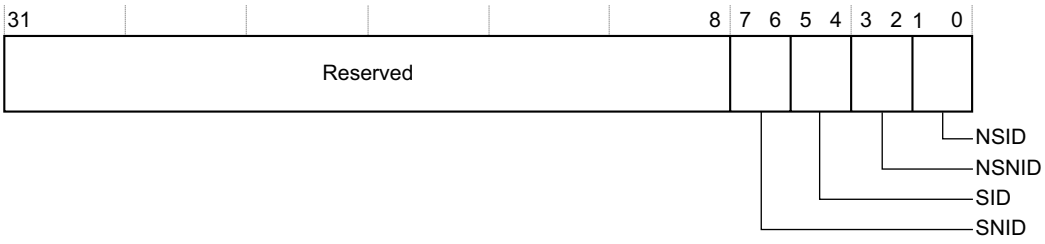


Figure 3-191 AUTHSTATUS Register bit assignments

Table 3-198 shows the bit assignments.

Table 3-198 AUTHSTATUS Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:6]	SNID	Indicates the security level for secure non-invasive debug: 0b00 Functionality is not implemented or is controlled elsewhere.
[5:4]	SID	Indicates the security level for secure invasive debug: 0b00 Functionality is not implemented or is controlled elsewhere.
[3:2]	NSNID	Indicates the security level for non-secure non-invasive debug: 0b00 Functionality is not implemented or is controlled elsewhere.
[1:0]	NSID	Indicates the security level for non-secure invasive debug: 0b00 Functionality is not implemented or is controlled elsewhere.

3.15.26 Device Configuration Register

The DEVID Register characteristics are:

Purpose Indicates the capabilities of the component.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-192 shows the bit assignments.

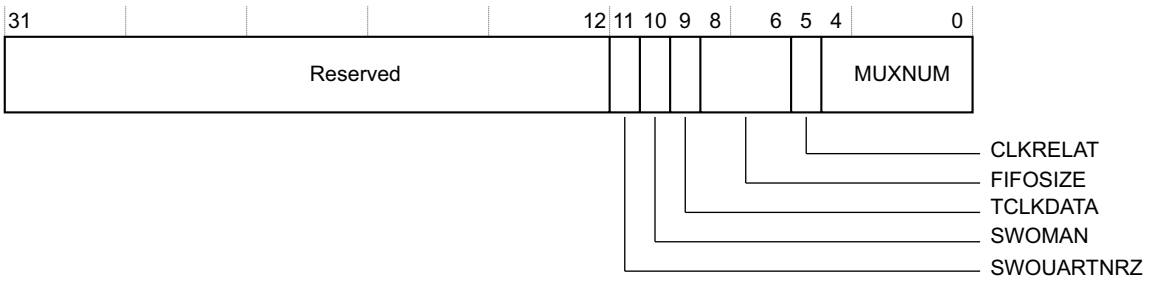


Figure 3-192 DEVID Register bit assignments

Table 3-199 shows the bit assignments.

Table 3-199 DEVID Register bit assignments

Bits	Name	Function
[31:12]	Reserved	-
[11]	SWOUARTNRZ	Indicates whether Serial Wire Output, UART or NRZ, is supported. **0** Serial Wire Output, UART or NRZ, is not supported.
[10]	SWOMAN	Indicates whether Serial Wire Output, Manchester encoded format, is supported. **0** Serial Wire Output, Manchester encoded format, is not supported.
[9]	TCLKDATA	Indicates whether trace clock plus data is supported. **0** Trace clock and data is supported.
[8:6]	FIFOSIZE	FIFO size in powers of 2. `0b010` FIFO size of 4 entries, that is, 16 bytes.
[5]	CLKRELAT	Indicates the relationship between **atclk** and **traceclkin**. **1** **atclk** and **traceclkin** are asynchronous.
[4:0]	MUXNUM	Indicates the hidden level of input multiplexing. When non-zero this value indicates the type or number of ATB multiplexing present on the input to the ATB. Currently only `0x00` is supported, that is, no multiplexing is present. This value is used to assist in topology detection of the ATB structure.

3.15.27 Device Type Identifier Register

The DEVTYPE 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 See the register summary in Table 3-173 on page 3-141.

Figure 3-193 shows the bit assignments.

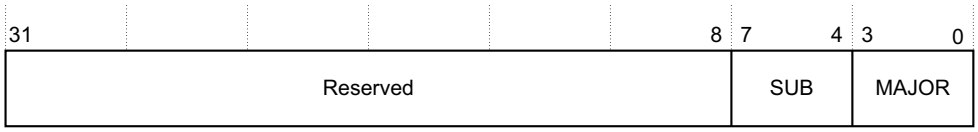


Figure 3-193 DEVTYPE Register bit assignments

Table 3-200 shows the bit assignments.

Table 3-200 DEVTYPE Register bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	SUB	Sub-classification of the type of the debug component as specified in the *CoreSight Architecture Specification* within the major classification as specified in the MAJOR field. `0b0001` Indicates that this component is a trace port component.
[3:0]	MAJOR	Major classification of the type of the debug component as specified in the *CoreSight Architecture Specification* for this debug and trace component. `0b0001` Indicates that this component is a trace sink component.

3.15.28 Peripheral ID4 Register

The PIDR4 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-194 shows the bit assignments.

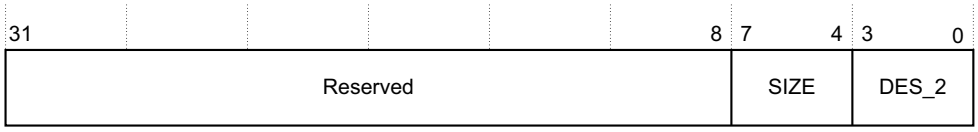


Figure 3-194 PIDR4 bit assignments

Table 3-201 shows the bit assignments.

Table 3-201 PIDR4 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	SIZE	This is a 4-bit value that indicates the total contiguous size of the memory window this component uses in powers of 2 from the standard 4KB. `0b0000` Indicates that the device only occupies 4KB of memory.
[3:0]	DES_2	This is the JEDEC JEP106 continuation code. This code, along with bits[6:4] of the identity code defined in the PIDR0 and bits[3:0] of the identity code defined in the PIDR1, identifies the designer of the component. `0b0100` JEDEC continuation code.

3.15.29 Peripheral ID5-7 registers

The PIDR5-7 characteristics are:

Purpose Reserved.

Usage constraints There are no usage constraints.

Configurations These registers are available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-195 shows the bit assignments.

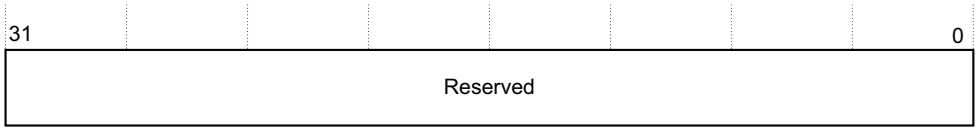


Figure 3-195 PIDR5-7 bit assignments

Table 3-202 shows the bit assignments.

Table 3-202 PIDR5-7 bit assignments

Bits	Name	Function
[31:0]	Reserved	-

3.15.30 Peripheral ID0 Register

The PIDR0 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 See the register summary in Table 3-173 on page 3-141.

Figure 3-196 shows the bit assignments.

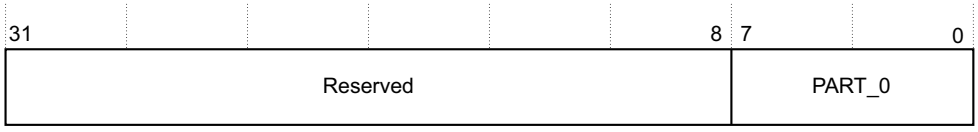


Figure 3-196 PIDR0 bit assignments

Table 3-203 shows the bit assignments.

Table 3-203 PIDR0 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PART_0	Bits[7:0] of the 12-bit part number of the component. The designer of the component assigns this part number. `0x12` Indicates bits[7:0] of the part number of the component.

3.15.31 Peripheral ID1 Register

The PIDR1 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 See the register summary in Table 3-173 on page 3-141.

Figure 3-197 shows the bit assignments.

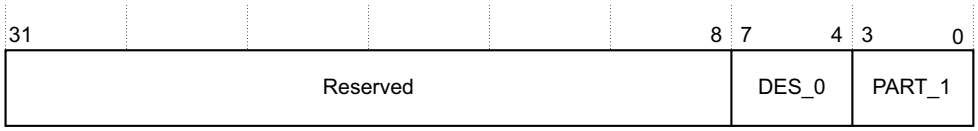


Figure 3-197 PIDR1 bit assignments

Table 3-204 shows the bit assignments.

Table 3-204 PIDR1 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	DES_0	Bits[3:0] of the JEDEC JEP106 identity code. This code, along with bits[6:4] of the identity code defined in the PIDR2 and the continuation code defined in the PIDR4, identifies the designer of the component. `0b1011` Bits[3:0] of the JEDEC JEP106 Identity Code. The default value is ARM.
[3:0]	PART_1	Bits[11:8] of the 12-bit part number of the component. The designer of the component assigns this part number. `0b1001` Indicates bits[11:8] of the part number of the component.

3.15.32 Peripheral ID2 Register

The PIDR2 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 See the register summary in Table 3-173 on page 3-141.

Figure 3-198 shows the bit assignments.

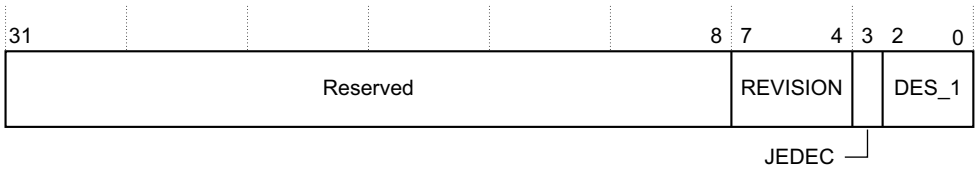


Figure 3-198 PIDR2 bit assignments

Table 3-205 shows the bit assignments.

Table 3-205 PIDR2 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	REVISION	An incremental value starting from `0b0000` for the first revision of this component. This increases by 1 for both major and minor revisions and is used to identify the major or minor revisions. `0b0100` This device is at r0p4.
[3]	JEDEC	Always set. Indicates whether the JEDEC assigned designer ID is used. `0b1` Indicates that a JEDEC assigned designer ID is used.
[2:0]	DES_1	Bits[6:4] of the JEDEC JEP106 identity code. This code, along with bits[3:0] of the identity code defined in the PIDR1 and the continuation code defined in the PIDR4, identifies the designer of the component. `0b011` Bits[6:4] of the JEDEC JEP106 Identity Code. The default value is ARM.

3.15.33 Peripheral ID3 Register

The PIDR3 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-199 shows the bit assignments.

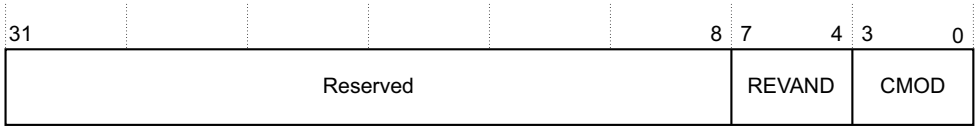


Figure 3-199 PIDR3 bit assignments

Table 3-206 shows the bit assignments.

Table 3-206 PIDR3 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	REVAND	Indicates minor errata fixes specific to the revision of the component being used, for example metal fixes after implementation. In most cases, this field is zero. ARM recommends that the component designers ensure that a metal fix can change this field if required, for example, by driving it from registers that reset to zero. `0b0000` Indicates that there are no errata fixes to this component.
[3:0]	CMOD	Indicates whether the customer has modified the behavior of the component. In most cases, this field is zero. The customer changes this value on modifications to this component. `0b0000` Indicates that the customer has not modified this component.

3.15.34 Component ID0 Register

The CIDR0 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-200 shows the bit assignments.

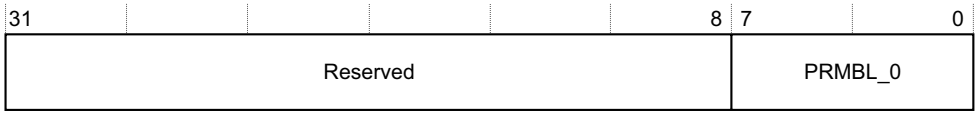


Figure 3-200 CIDR0 bit assignments

Table 3-207 shows the bit assignments.

Table 3-207 CIDR0 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_0	Contains bits[7:0] of the component identification code. `0x0D` Bits[7:0] of the identification code.

3.15.35 Component ID1 Register

The CIDR1 characteristics are:

Purpose A component identification register that indicates 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 See the register summary in Table 3-173 on page 3-141.

Figure 3-201 shows the bit assignments.

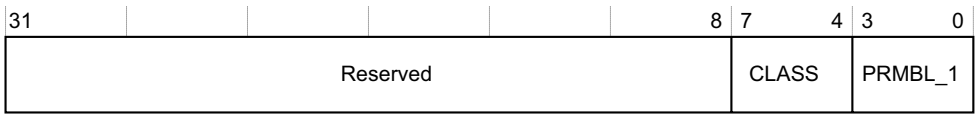


Figure 3-201 CIDR1 bit assignments

Table 3-208 shows the bit assignments.

Table 3-208 CIDR1 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	CLASS	Class of the component, for example, if the component is a ROM table or a generic CoreSight component. Contains bits[15:12] of the component identification code. `0b1001` Indicates that the component is a CoreSight component.
[3:0]	PRMBL_1	Contains bits[11:8] of the component identification code. `0b0000` Bits[11:8] of the identification code.

3.15.36 Component ID2 Register

The CIDR2 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 See the register summary in Table 3-173 on page 3-141.

Figure 3-202 shows the bit assignments.

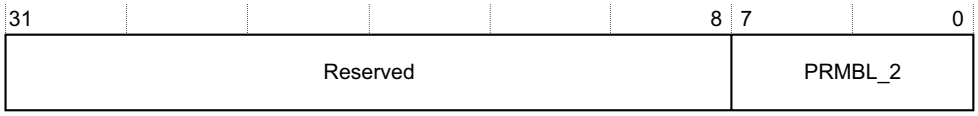


Figure 3-202 CIDR2 bit assignments

Table 3-209 shows the bit assignments.

Table 3-209 CIDR2 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_2	Contains bits[23:16] of the component identification code. `0x05` Bits[23:16] of the identification code.

3.15.37 Component ID3 Register

The CIDR3 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-173 on page 3-141.

Figure 3-203 shows the bit assignments.

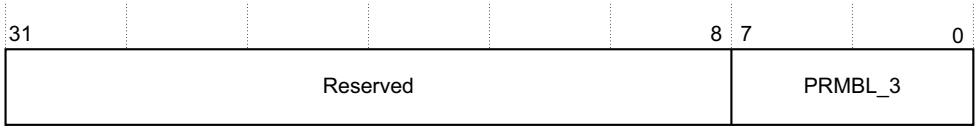


Figure 3-203 CIDR3 bit assignments

Table 3-210 shows the bit assignments.

Table 3-210 CIDR3 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_3	Contains bits[31:24] of the component identification code. `0xB1` Bits[31:24] of the identification code.

3.16 DAP register summary

This section shows the following DAP components register summary:

- *JTAG-AP register summary*.
- *AHB-AP register summary*.
- *AXI-AP register summary* on page 3-181.
- *APB-AP register summary* on page 3-181.
- *Debug port register summary* on page 3-182.

3.16.1 JTAG-AP register summary

Table 3-211 shows the JTAG-AP register summary.

Table 3-211 JTAG-AP register summary

Offset	Type	Reset value	Name
0x00	RW	0x00000000	*JTAG-AP Control/Status Word Register, CSW, 0x00* on page 3-184, CSW.
0x04	RW	0x00	*JTAG-AP Port Select Register, PORTSEL, 0x04* on page 3-185, PORTSEL.
0x08	RW	0x00	*JTAG-AP Port Status Register, PSTA, 0x08* on page 3-185, PSTA.
0x0C	-	-	Reserved.
0x10	RW	UNDEFINED	*JTAG-AP Byte FIFO registers, BFIFOn, 0x10-0x1C* on page 3-186.
0x14	RW	UNDEFINED	
0x18	RW	UNDEFINED	
0x1C	RW	UNDEFINED	
0x20-0xF8	-	-	Reserved, SBZ.
0xFC	RO	0x24760010	*JTAG-AP Identification Register* on page 3-186, IDR. Required by all access ports.

3.16.2 AHB-AP register summary

Table 3-212 shows the AHB-AP register summary.

Table 3-212 AHB-AP register summary

Offset	Type	Reset value	Name
0x00	RW	0x40000002	*AHB-AP Control/Status Word Register, CSW, 0x00* on page 3-187, CSW.
0x04	RW	0x00000000	*AHB-AP Transfer Address Register, TAR, 0x04* on page 3-188, TAR.
0x08	-	-	Reserved, SBZ.
0x0C	RW	-	*AHB-AP Data Read/Write Register, DRW, 0x0C* on page 3-189, DRW.
0x10	RW	-	*AHB-AP Banked Data Registers, BD0-BD03, 0x10-0x1C* on page 3-189.
0x14	RW	-	
0x18	RW	-	
0x1C	RW	-	

Table 3-212 AHB-AP register summary (continued)

Offset	Type	Reset value	Name
0x20-0xF7	-	-	Reserved, SBZ.
0xF8	RO	IMPLEMENTATION DEFINED	*ROM Address Register, ROM, 0xF8* on page 3-189.
0xFC	RO	0x64770001	*AHB-AP Identification Register, IDR, 0xFC* on page 3-189, IDR.

3.16.3 AXI-AP register summary

Table 3-213 shows the AXI-AP register summary.

Table 3-213 AXI-AP register summary

Offset	Type	Reset value	Name
0x00	RW	-	*AXI-AP Control/Status Word Register* on page 3-190, CSW.
0x04	RW	-	*AXI-AP Transfer Address Register* on page 3-192, TAR.
0x08	RW	-	
0x0C	RW	-	*AXI-AP Data RW Register* on page 3-193, DRW.
0x10	RW	-	*AXI-AP Banked Data Registers* on page 3-193.
0x14	RW	-	
0x18	RW	-	
0x1C	RW	-	
0x20	RW	-	*AXI-AP ACE Barrier Transaction* on page 3-194.
0x24-0xEC	-	-	Reserved, SBZ.
0xF0	RO	-	*AXI-AP Debug Base Address Register* on page 3-195.
0xF8	RO	-	
0xF4	RO	-	*AXI-AP Configuration Register* on page 3-195.
0xFC	RO	-	*AXI-AP Identification Register* on page 3-196.

3.16.4 APB-AP register summary

Table 3-214 shows the APB-AP register summary.

Table 3-214 APB-AP register summary

Offset	Type	Reset value	Name
0x00	RW	0x00000002	*APB-AP Control/Status Word Register, CSW, 0x00* on page 3-197, CSW.
0x04	RW	0x00000000	*APB-AP Transfer Address Register, TAR, 0x04* on page 3-198, TAR.
0x08	-	-	Reserved, SBZ.
0x0C	RW	-	*APB-AP Data Read/Write Register, DRW, 0x0C* on page 3-199, DRW.

Table 3-214 APB-AP register summary (continued)

Offset	Type	Reset value	Name
0x10	RW	-	*APB-AP Banked Data Registers, BD0-BD3, 0x10-0x1C* on page 3-199.
0x14	RW	-	
0x18	RW	-	
0x1C	RW	-	
0x20-0xF4	-	-	Reserved, SBZ.
0xF8	RO	0x80000000	*Debug APB ROM Address, ROM, 0xF8* on page 3-199, ROM.
0xFC	RO	0x44770002	*APB-AP Identification Register* on page 3-200, IDR.

3.16.5 Debug port register summary

Table 3-215 shows the DP register summary, and summarizes which registers are implemented on a JTAG-DP and which are implemented on a SW-DP.

Table 3-215 Debug port register summary

Name	JTAG-DP	SW-DP	Description
ABORT	Yes	Yes	AP Abort Register. See *AP Abort Register, ABORT* on page 3-200.
IDCODE	Yes	Yes	ID Code Register. See *Identification Code Register, IDCODE* on page 3-201.
CTRL/STAT	Yes	Yes	DP Control/Status Register. See *Control/Status Register, CTRL/STAT* on page 3-202.
SELECT	Yes	Yes	Select Register. See *AP Select Register, SELECT* on page 3-204.
RDBUFF	Yes	Yes	Read Buffer. See *Read Buffer, RDBUFF* on page 3-206.
WCR	No	Yes	Wire Control Register. See *Wire Control Register, WCR (SW-DP only)* on page 3-206.
TARGETID	No	Yes	Target Identification Register. See *Target Identification Register, TARGETID (SW-DP only)* on page 3-208.
DLPIDR	No	Yes	Data Link Protocol Identification Register. See *Data Link Protocol Identification Register, DLPIDR (SW-DP only)* on page 3-208.
RESEND	No	Yes	Read Resend Register. See *Read Resend Register, RESEND (SW-DP only)* on page 3-209.

JTAG-DP register summary

Table 3-216 on page 3-183 shows all implemented registers accessible through the JTAG interface. All other *Instruction Register* (IR) instructions are implemented as BYPASS, and an external TAP controller must be implemented in accordance with the *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*, if more IR registers are required, for example, JTAG TAP boundary scan. See *JTAP-DP register descriptions* on page 3-209.

Table 3-216 JTAG-DP register summary

IR instruction value	JTAG-DP register	DR scan width	Description
0b1000	ABORT	35	JTAG-DP Abort Register, ABORT.
0b1010	DPACC	35	JTAG DP/AP Access Registers, DPACC/APACC.
0b1011	APACC	35	
0b1110	IDCODE	32	JTAG Device ID Code Register, IDCODE.
0b1111	BYPASS	1	JTAG Bypass Register, BYPASS.

3.17 DAP register descriptions

This section describes the following DAP components registers and their bit assignments:

- *JTAG-AP register descriptions*.
- *AHB-AP register descriptions* on page 3-187.
- *AXI-AP register descriptions* on page 3-190.
- *APB-AP register descriptions* on page 3-196.
- *Debug port implementation-specific registers* on page 3-200.

3.17.1 JTAG-AP register descriptions

All the registers are described in *ARM® Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*.

JTAG-AP Control/Status Word Register, CSW, 0x00

Purpose Used to configure and control transfers through the JTAG interface.

Attributes See *DAP register summary* on page 3-180 for more information

Figure 3-204 shows the bit assignments.

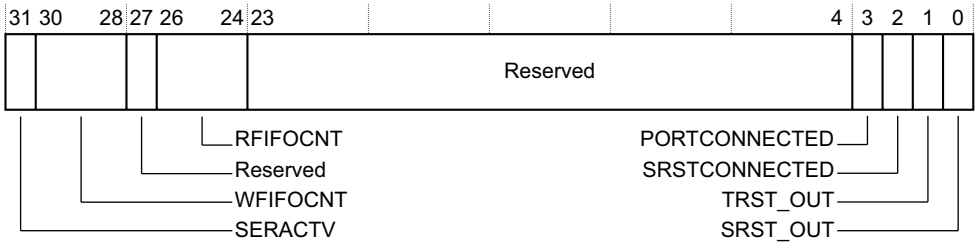


Figure 3-204 JTAG-AP CSW Register bit assignments

Table 3-217 shows the bit assignments. The register must not be modified while there are outstanding commands in the Write FIFO.

Table 3-217 JTAG-AP CSW Register bit assignments

Bits	Type	Name	Function
[31]	RO	SERACTV	JTAG serializer active. The reset value is `0b0`.
[30:28]	RO	WFIFOCNT	Outstanding write FIFO byte count. The reset value is `0b000`.
[27]	-	-	Reserved, SBZ.
[26:24]	RO	RFIFOCNT	Outstanding read FIFO byte count. The reset value is `0b000`.
[23:4]	-	-	Reserved, SBZ.
[3]	RO	PORTCONNECTED	PORT connected. AND of **portconnected** inputs of currently selected ports. The reset value is `0b0`.

Table 3-217 JTAG-AP CSW Register bit assignments (continued)

Bits	Type	Name	Function
[2]	RO	SRSTCONNECTED[a]	SRST connected. AND of **srstconnected** inputs of currently selected ports. If multiple ports are selected, it is the AND of all the **srstconnected** inputs from the selected ports. The reset value is `0b0`.
[1]	RW	TRST_OUT	TRST assert, not self clearing. The JTAG TAP controller reset. The reset value is `0b0`.
[0]	RW	SRST_OUT	SRST assert, not self clearing. Core reset. The reset value is `0b0`.

a. **SRSTCONNECTED** is a strap pin on the multiplexer inputs. It is set to 1 to indicate that the target JTAG device supports individual SRST controls.

JTAG-AP Port Select Register, PORTSEL, 0x04

Purpose Enables ports if connected and the slave port is currently enabled. The Port Select Register must be written when the TCK engine is idle, **SERACTV**=0, and **WFIFO**, **WFIFOCNT**=0, is empty. Writing at other times can generate unpredictable results.

Attributes See *DAP register summary* on page 3-180 for more information

Figure 3-205 shows the bit assignments.

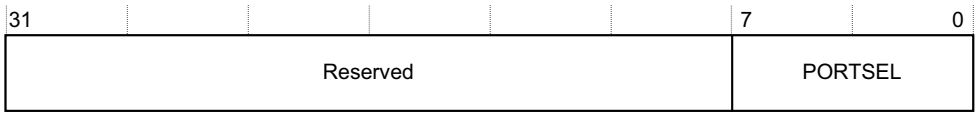


Figure 3-205 JTAG-AP Port Select Register bit assignments

Table 3-218 shows the bit assignments.

Table 3-218 JTAG-AP Port Select Register bit assignments

Bits	Type	Name	Function
[31:8]	-	-	Reserved SBZ.
[7:0]	RW	PORTSEL	Port Select. The reset value is `0x00`.

JTAG-AP Port Status Register, PSTA, 0x08

Purpose A sticky register that captures the state of a connected and selected port on every clock cycle. If a connected and selected port is disabled or powered down, even transiently, the corresponding bit in the Port Status Register is set. It remains set until cleared by writing a one to the corresponding bit.

Attributes See *DAP register summary* on page 3-180 for more information.

Figure 3-206 on page 3-186 shows the bit assignments.

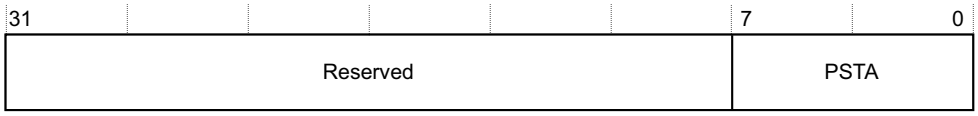


Figure 3-206 JTAG-AP Port Status Register bit assignments

Table 3-219 shows the bit assignments.

Table 3-219 JTAG-AP Port Status Register bit assignments

Bits	Type	Name	Function
[31:8]	-	-	Reserved, SBZ.
[7:0]	RW	PSTA	Port Status. The reset value is `0x00`.

JTAG-AP Byte FIFO registers, BFIFOn, 0x10-0x1C

Purpose Word interface to one, two, three, or four parallel byte entries in the byte command FIFO, LSB first. The DAP internal bus is a 32-bit interface with no SIZE field. So, an address decoding is used to designate size, because the JTAG-AP engine JTAG protocol is byte encoded. Writes to the BFIFOx larger than the current write FIFO depth stall on **dapready** in normal mode. Reads to the BFIFOx larger than the current read FIFO depth stall on **dapready** in normal mode. For reads less than the full 32-bits, the upper bits are zero. For example, for a 24-bit read, **daprdata[31:24]** is `0x00`.

Attributes See *DAP register summary* on page 3-180 for more information.

JTAG-AP Identification Register

Figure 3-207 shows the bit assignments.

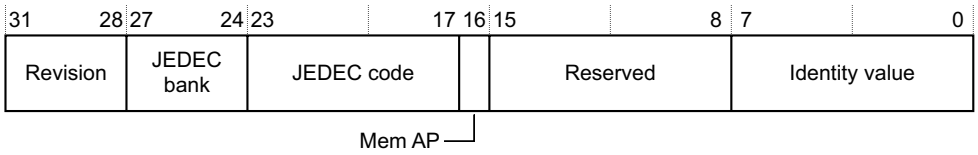


Figure 3-207 JTAG-AP Identification Register bit assignments

Table 3-220 shows the bit assignments.

Table 3-220 JTAG-AP Identification Register bit assignments

Bits	Type	Name	Value	Function
[31:28]	RO	Revision	`0x2`	Revision 2
[27:24]	RO	JEDEC bank	`0x4`	Designed by ARM
[23:17]	RO	JEDEC code	`0x3B`	Designed by ARM
[16]	RO	Mem AP	`0x0`	Is a Mem AP
[15:8]	-	Reserved	`0x00`	-
[7:0]	RO	Identity value	`0x10`	JTAG-AP

3.17.2 AHB-AP register descriptions

This section describes the registers used to program the AHB-AP. It contains the following registers:

- *AHB-AP Control/Status Word Register, CSW, 0x00*.
- *AHB-AP Transfer Address Register, TAR, 0x04* on page 3-188.
- *AHB-AP Data Read/Write Register, DRW, 0x0C* on page 3-189.
- *AHB-AP Banked Data Registers, BD0-BD03, 0x10-0x1C* on page 3-189.
- *ROM Address Register, ROM, 0xF8* on page 3-189.
- *AHB-AP Identification Register, IDR, 0xFC* on page 3-189.

AHB-AP Control/Status Word Register, CSW, 0x00

Purpose Used to configure and control transfers through the AHB interface.

Attributes See *DAP register summary* on page 3-180 for more information.

Figure 3-208 shows the bit assignments.

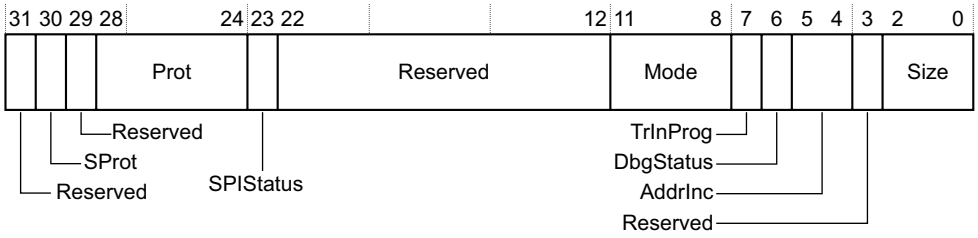


Figure 3-208 AHB-AP CSW Register bit assignments

Table 3-221 shows the bit assignments.

Table 3-221 AHB-AP Control/Status Word Register bit assignments

Bits	Type	Name	Function
[31]	-	-	Reserved SBZ.
[30]	RW	SProt	Specifies that a secure transfer is requested. SProt HIGH indicates a non-secure transfer. **SProt** LOW indicates a secure transfer. • If this bit is LOW, and **spiden** is HIGH, **hprot[6]** is asserted LOW on an AHB transfer. • If this bit is LOW, and **spiden** is LOW, **hprot[6]** is asserted HIGH and the AHB transfer is not initiated. • If this bit is HIGH, the state of **spiden** is ignored. **hprot[6]** is HIGH. The reset value is `0b1`. This field is non-secure.
[29]	-	-	Reserved, SBZ.
[28:24]	RW	Prot	Specifies the protection signal encoding to be output on **hprot[4:0]**. The reset value is `0b00011`. This field is non secure, non-exclusive, non cacheable, non bufferable, data access, and privileged.
[23]	RO	SPIStatus	Indicates the status of the **spiden** port. If SPIStatus is LOW, no secure AHB transfers are carried out.
[22:12]	-	-	Reserved, SBZ.

Table 3-221 AHB-AP Control/Status Word Register bit assignments (continued)

Bits	Type	Name	Function
[11:8]	RW	Mode	Specifies the mode of operation. `0b0000` Normal download or upload model `0b0001-0b1111` Reserved, SBZ. The reset value is `0b0000`.
[7]	RO	TrInProg	Transfer in progress. This field indicates whether a transfer is currently in progress on the AHB master port.
[6]	RO	DbgStatus	Indicates the status of the **dbgen** port. If DbgStatus is LOW, no AHB transfers are carried out. **1** AHB transfers permitted. **0** AHB transfers not permitted.
[5:4]	RW	AddrInc	Auto address increment and packing mode on RW data access. Only increments if the current transaction completes without an error response and the transaction is not aborted. Auto address incrementing and packed transfers are not performed on access to Banked Data registers, `0x10-0x1C`. The status of these bits is ignored in these cases. Increments and wraps within a 1KB address boundary, for example, for word incrementing from `0x1400-0x17FC`. If the start is at `0x14A0`, then the counter increments to `0x17FC`, wraps to `0x1400`, then continues incrementing to `0x149C`. `0b00` Auto increment OFF. `0b00` Increment, single. Single transfer from corresponding byte lane. `0b10` Increment, packed. `Word` Same effect as single increment. `Byte or halfword` Packs four 8-bit transfers or two 16-bit transfers into a 32-bit DAP transfer. Multiple transactions are carried out on the AHB interface. `0b11` Reserved, SBZ. No transfer. Size of address increment is defined by the Size field, bits [2:0]. The reset value is `0b00`.
[3]	RW	-	Reserved, SBZ. The reset value is `0b0`.
[2:0]	RW	Size	Size of the data access to perform. `0b000` 8 bits. `0b001` 16 bits. `0b010` 32 bits. `0b011-0b111` Reserved, SBZ. The reset value is `0b010`.

AHB-AP Transfer Address Register, TAR, 0x04

Table 3-222 shows the bit assignments.

Table 3-222 AHB-AP Transfer Address Register bit assignments

Bits	Type	Name	Function
[31:0]	RW	Address	Address of the current transfer. The reset value is `0x00000000`.

AHB-AP Data Read/Write Register, DRW, 0x0C

Table 3-223 shows the bit assignments.

Table 3-223 AHB-AP Data Read/Write Register bit assignments

Bits	Type	Name	Function	
[31:0]	RW	Data	**Write mode**	Data value to write for the current transfer.
			Read mode	Data value read from the current transfer.

AHB-AP Banked Data Registers, BD0-BD03, 0x10-0x1C

Purpose BD0-BD3 provide a mechanism for directly mapping through DAP accesses to AHB transfers without having to rewrite the *Transfer Address Register* (TAR) within a four-location boundary. BD0 RW from TA. BD1 RW from TA+4.

Attributes See *DAP register summary* on page 3-180 for more information.

Table 3-224 shows the bit assignments.

Table 3-224 Banked Data Register bit assignments

Bits	Type	Name	Function
[31:0]	RW	Data	If **dapcaddr[7:4]** = `0x0001`, it is accessing AHB-AP registers in the range `0x10-0x1C`, and the derived **haddr[31:0]** is: **Write mode** Data value to write for the current transfer to external address TAR[31:4] + **dapcaddr[3:2]** + `0b00`. **Read mode** Data value read from the current transfer from external address TAR[31:4] + **dapcaddr[3:2]** + `0b00`. Auto address incrementing is not performed on DAP accesses to BD0-BD3. Banked transfers are only supported for word transfers. Non-word banked transfers are reserved and UNPREDICTABLE. Transfer size is currently ignored for banked transfers.

ROM Address Register, ROM, 0xF8

Table 3-225 shows the bit assignments.

Table 3-225 ROM Address Register bit assignments

Bits	Type	Name	Function
[31:0]	RO	Debug AHB ROM Address	Base address of a ROM table. The ROM provides a look-up table for system components. Set to the tie-off value on the static input port, **rombaseaddr**.

AHB-AP Identification Register, IDR, 0xFC

Figure 3-209 on page 3-190 shows the bit assignments.

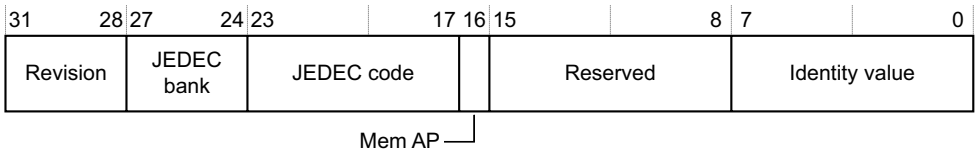


Figure 3-209 AHB-AP Identification Register bit assignments

Table 3-226 shows the bit assignments.

Table 3-226 AHB-AP Identification Register bit assignments

Bits	Type	Name	Value	Function
[31:28]	RO	Revision	0x6	Revision 6
[27:24]	RO	JEDEC bank	0x4	Designed by ARM
[23:17]	RO	JEDEC code	0x3B	Designed by ARM
[16]	RO	Mem AP	0x1	Is a Mem AP
[15:8]	-	Reserved	0x00	-
[7:0]	RO	Identity value	0x01	AHB-AP

3.17.3 AXI-AP register descriptions

This section describes the following AXI-AP registers:

- *AXI-AP Control/Status Word Register*.
- *AXI-AP Transfer Address Register* on page 3-192.
- *AXI-AP Data RW Register* on page 3-193.
- *AXI-AP Banked Data Registers* on page 3-193.
- *AXI-AP ACE Barrier Transaction* on page 3-194.
- *AXI-AP Debug Base Address Register* on page 3-195.
- *AXI-AP Configuration Register* on page 3-195.
- *AXI-AP Identification Register* on page 3-196.

AXI-AP Control/Status Word Register

Purpose Used to configure and control transfers through the AXI interface.

Attributes See *DAP register summary* on page 3-180 for more information.

Figure 3-210 on page 3-191 shows the bit assignments.

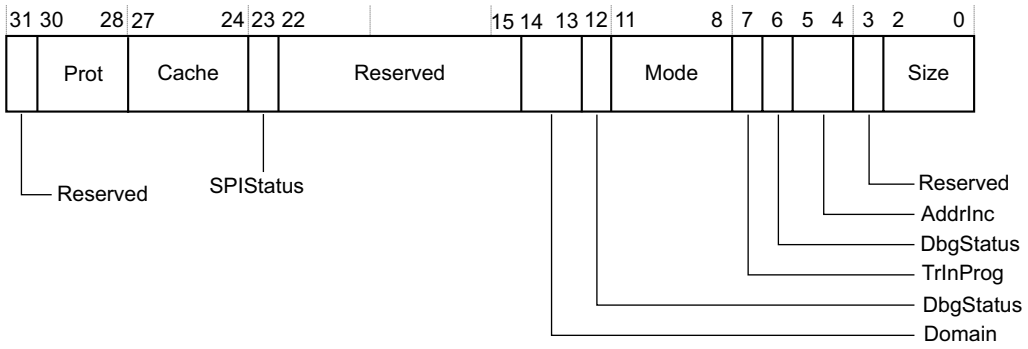


Figure 3-210 AXI-AP CSW Register bit assignments

Table 3-227 shows the bit assignments.

Table 3-227 AXI-AP CSW Register bit assignments

Bits	Type	Name	Reset value	Function
[31]	-	Reserved	-	-
[30:28]	RW	Prot	`0b011`	Specifies protection encoding as AMBA AXI protocol describes.
[27:24]	RW	Cache	`0b0000`	Specifies the cache encoding as AMBA AXI protocol describes.
[23]	RO	SPIStatus	-	Indicates the status of the **spiden** port. If **SPIStatus** is LOW, then no secure AXI transfers are carried out.
[22:15]	-	Reserved	-	-
[14:13]	RW	Domain	`0b00`	Shareable transaction encoding for ACE. `0b00` Non shareable. `0b01` Shareable, inner domain includes additional masters. `0b10` Shareable, outer domain, also includes inner or additional master `0b11` Shareable, system domain, all masters included.
[12]	RW	ACEEnable	`0b0`	Enable ACE transactions, including barriers. `0b0` Disable. `0b1` Enable.
[11:8]	RW	Mode	`0b0000`	Specifies the mode of operation: `0b0000` Normal download or upload. `0b0001` Barrier transaction. `0b0010-0b1111` Reserved, SBZ.
[7]	RO	TrInProg	-	Transfer in progress. This field indicates whether a transfer is currently in progress on the AXI master port.
[6]	RO	DbgStatus		Indicates the status of DBGEN port. If **DbgStatus** is LOW, then no AXI transfers are carried out. `0b0` AXI transactions are stopped. `0b1` AXI transactions are permitted.

Table 3-227 AXI-AP CSW Register bit assignments (continued)

Bits	Type	Name	Reset value	Function
[5:4]	RW	AddrInc	0b00	Auto address increment and packing mode on RW data access. Only increments if the current transaction completes without an Error response and the transaction is not aborted. Auto address incrementing and packed data transfers are not performed on access to banked data registers 0x10-0x1C. The status of these bits is ignored in these cases. Following are the increments and wraps within a 1K address boundary: 0b00 Auto increment OFF. 0b01 Single increment. Single transfer from byte lane. 0b10 Increment packed. **Word** Same effect as single increment. **Byte or Halfword** Packs of four 8-bit transfers or two 16-bit transfers into a 32-bit DAP transfer. 0b11 Reserved, no transfer. The size of address increment is defined by the Size field.
[3]	-	Reserved		-
[2:0]	RW	Size	0b010	Size of the data access to perform. 0b000 8-bit. 0b001 16-bit. 0b010 32-bit. 0b011 64-bit. 0b100-0b111 Reserved, SBZ.

AXI-AP Transfer Address Register

Purpose Defines the current address of the transfer. In case of a 32-bit address, this contains the entire address value. In case of LPAE, this contains only the lower 32-bit of the address.

Attributes See *DAP register summary* on page 3-180 for more information.

Figure 3-211 shows the bit assignments.

Figure 3-211 AXI-AP Transfer Address Register bit assignments

Table 3-228 shows the bit assignments.

Table 3-228 AXI-AP Transfer Address Register bit assignments

Bits	Type	Name	Reset value	Function
[63:32]	RW	Address	0x00000000	Address of the current transfer.
[31:0]	RW	Address	0x00000000	Address of the current transfer.

AXI-AP Data RW Register

Purpose Stores the read data to be read in case of a read transfer. In case of a write transfer, write data must be written in the register.

Attributes See *DAP register summary* on page 3-180 for more information.

Figure 3-212 shows the bit assignments.

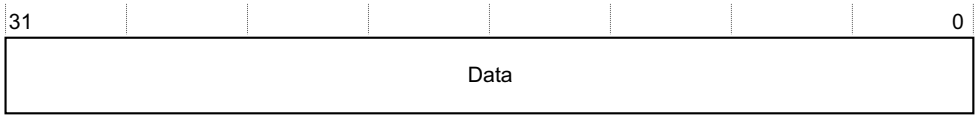


Figure 3-212 AXI-AP Data RW Register bit assignments

Table 3-229 shows the bit assignments.

Table 3-229 AXI-AP Data RW Register bit assignments

Bits	Type	Name	Function
[31:0]	RW	Data	In case of a 32-bit data access on the AXI-interface, store or write the 32-bits of data into this register once. Following are the modes for this register: **Read mode** Data value read from the current transfer. **Write mode** Data value to write for the current transfer.

Note

In case of a 64-bit access, multiple accesses must be initiated to DRW to make a single AXI access.

Read The first read of DRW in a sequence initiates a memory access. The first read returns the lower 32-bits of data. Subsequent read access returns the upper 32-bits of data. If a write to the CSW or TAR is initiated before the sequence completes, then the read access is terminated, and read data is no longer available.

Write The first write to DRW specifies the lower 32-bits of data to be written. Subsequent write access specifies the upper 32-bits to be written. If a write to the CSW is initiated before the sequence completes, then the write access is not initiated on the AXI interface.

- combining partial reads and writes within a sequence terminates the earlier access and the latest access is not recognized
- any write access to CSW Register, TAR Register, or to any other register in the AP during a sequence terminates the ongoing access and the current access is not recognized
- if a write sequence is terminated, then there is no write on AXI interface.

AXI-AP Banked Data Registers

Purpose BD0-3 provide a mechanism for direct mapping through DAP accesses to AXI transfers without having to rewrite the Transfer Address Register within a 4-location boundary. For example, BD0 reads and writes from TAR. BD1 reads and writes from TAR+4. This is applicable for a 32-bit access.

Attributes See *DAP register summary* on page 3-180 for more information.

Figure 3-213 on page 3-194 shows the bit assignments.

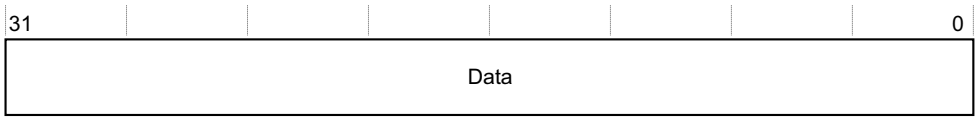


Figure 3-213 AXI-AP Banked DATA Register bit assignments

Table 3-230 shows the bit assignments.

Table 3-230 AXI-AP Banked Data Registers bit assignments

Bits	Type	Name	Function
[31:0]	RW	Data	If **dapcaddr[7:4]** = `0x0001`, it is accessing AXI-AP registers in the range `0x10-0x1C`, and the derived `ADDR[ADDR_WIDTH-1:0]` in both RW, 32-bit mode is as follows: • For a 32-bit address mode, the external address is TAR[31:4] + DAPADDR[3:2] + `0b00`. • For the LPAE mode, the external address is TAR[63:4] + DAPADDR[3:2] + `0b00`. Auto address incrementing is not performed on DAP accesses to BD0-BD3. Banked transfers are only supported for word transfers in case of 32-bit data. Non-word banked transfers are reserved and UNPREDICTABLE. Transfer size is ignored for banked transfers.

AXI-AP ACE Barrier Transaction

Purpose Enables or disables the ACE barrier transactions.

Attributes See *DAP register summary* on page 3-180 for more information.

Figure 3-214 shows the bit assignments.

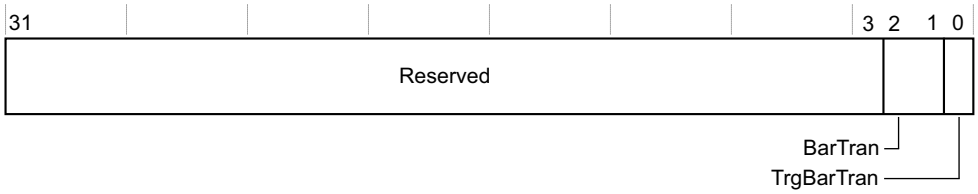


Figure 3-214 ACE Barrier Transaction Register bit assignments

Table 3-231 shows the bit assignments.

Table 3-231 ACE Barrier Transaction Register bit assignments

Bits	Type	Name	Reset value	Function
[31:3]	-	Reserved		-
[2:1]	RW	BarTran	`0b00`	Barrier transactions. `0b00` Normal access respecting barriers. `0b01` Memory barrier. `0b10` Reserved. `0b11` Synchronization barrier.
[0]	RW	TrgBarTran	`0b0`	The possible values are: **0** Disable barrier transaction. **1** Enable barrier transaction.

AXI-AP Debug Base Address Register

Purpose Provides an index into the connected memory-mapped resource. It points to one of these resources:

- The start of a set of debug registers.
- The ROM table that describes the connected debug component.

When the long address extension is implemented, the Debug Base Address Register is:

- A 64-bit register.
- It is split between offsets `0xF0` and `0xF8` in the register space.
- It is the first and the third register in the last register bank `0xF`:
  - BASE[31:0] are at offset `0xF8`.
  - BASE[63:32] are at offset `0xF0`.

Attributes See *DAP register summary* on page 3-180 for more information.

Figure 3-215 shows the bit assignments.

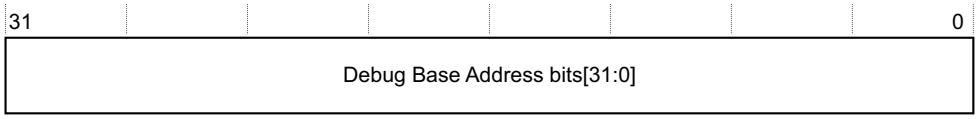


Figure 3-215 AXI-AP Debug Base Address Register bit assignments

Table 3-232 shows the bit assignments.

Table 3-232 AXI-AP Debug Base Address Register bit assignments

Bits	Type	Name	Function
[63:32]	RO	Debug Base Address bits [63:32]	Base address of either debug ROM table or start of a set of debug registers. The ROM table provides a look-up table for system components.
[31:0]	RO	Debug Base Address bits [31:0]	Base address of either debug ROM table or start of a set of debug registers. The ROM table provides a look-up table for system components.

AXI-AP Configuration Register

Purpose Provides information about revision.

Attributes See *DAP register summary* on page 3-180 for more information.

Figure 3-216 shows the bit assignments.

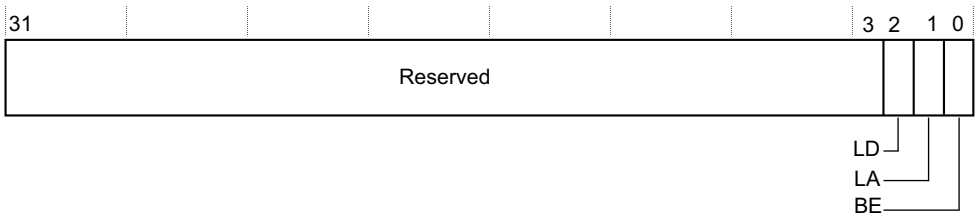


Figure 3-216 AXI-AP Configuration Register bit assignments

Table 3-233 shows the bit assignments.

Table 3-233 AXI-AP Configuration Register bit assignments

Bits	Type	Name	Function
[31:3]	-	Reserved	-
[2]	RO	LD	Large data. Indicates support for data items larger then 32-bits. **0** Only 8, 16, 32-bit data items are supported. **1** Support for 64-bit data item in addition to 8, 16, 32-bit data.
[1]	RO	LA	Long address. Indicates support for greater than 32-bits of addressing. **0** 32 or fewer bits of addressing. Registers `0x08` and `0xF0` are reserved. **1** 64 or fewer bits of addressing. The Transfer Address Register l and Debug Base Address Register l occupy two locations, at `0x04` and `0x08`, and at `0xF8` and `0xF0` respectively.
[0]	RO	BE	Big-endian. Always read as 0, because AXI-AP supports little-endian.

AXI-AP Identification Register

Purpose Provides information about revision.

Attributes See *DAP register summary* on page 3-180 for more information.

Figure 3-217 shows the bit assignments.

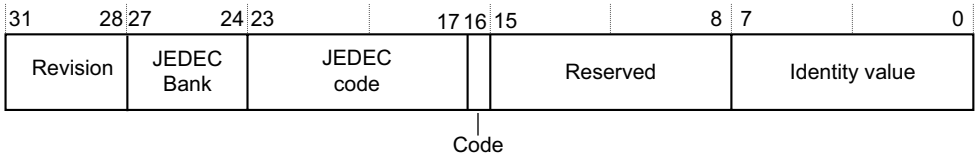


Figure 3-217 AXI-AP Identification Register bit assignments

Table 3-234 shows bit assignments.

Table 3-234 AXI-AP Identification Register bit assignments

Bits	Type	Name	Reset value	Function
[31:28]	RO	Revision	`0x1`	Revision value.
[27:24]	RO	JEDEC Bank	`0x4`	Designed by ARM.
[23:17]	RO	JEDEC Code	`0x3B`	Designed by ARM.
[16]	RO	Mem AP	`0x1`	Mem AP.
[15:8]	-	Reserved	`0x00`	-
[7:0]	RO	Identity value	`0x04`	AXI-AP.

3.17.4 APB-AP register descriptions

This section describes the following APB-AP registers:

- *APB-AP Control/Status Word Register, CSW, 0x00* on page 3-197.
- *APB-AP Transfer Address Register, TAR, 0x04* on page 3-198.
- *APB-AP Data Read/Write Register, DRW, 0x0C* on page 3-199.

- *APB-AP Banked Data Registers, BD0-BD3, 0x10-0x1C* on page 3-199.
- *Debug APB ROM Address, ROM, 0xF8* on page 3-199.
- *APB-AP Identification Register* on page 3-200.

APB-AP Control/Status Word Register, CSW, 0x00

Purpose Used to configure and control transfers through the APB interface.

Attributes See *DAP register summary* on page 3-180 for more information.

Figure 3-218 shows the bit assignments.

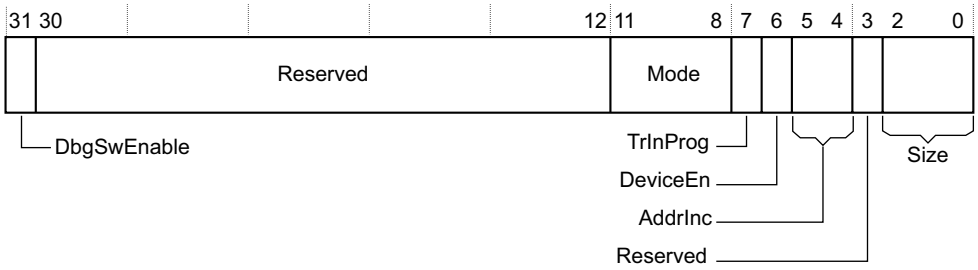


Figure 3-218 APB-AP Control/Status Word Register bit assignments

Table 3-235 shows the bit assignments.

Table 3-235 APB Control/Status Word Register bit assignments

Bits	Type	Name	Function
[31]	RW	DbgSwEnable	Software access enable. Drives **pdbgswen** to enable or disable software access to the Debug APB bus in the APB interconnect. **0** Disable software access. **1** Enable software access. The reset value is `0b0`. On exit from reset, defaults value is `b1` to enable software access.
[30:12]	-	-	Reserved, SBZ.
[11:8]	RW	Mode	Specifies the mode of operation. `0b0000` Normal download or upload model. `0b0001-0b1111` Reserved, SBZ. The reset value is `0b0000`.
[7]	RO	TrInProg	Transfer in progress. This field indicates whether a transfer is currently in progress on the APB master port.
[6]	RO	DeviceEn	Indicates the status of the **deviceen** input. • If APB-AP is connected to the Debug APB, a bus connected only to debug and trace components, it must be permanently enabled by tying **deviceen** HIGH. This ensures that trace components can still be programmed when **dbgen** is LOW. In practice, the APB-AP is normally used in this way. • If APB-AP is connected to a system APB dedicated to the non-secure world, **deviceen** must be connected to **dbgen**. • If APB-AP is connected to a system APB dedicated to the secure world, **deviceen** must be connected to **spiden**.

Table 3-235 APB Control/Status Word Register bit assignments (continued)

Bits	Type	Name	Function
[5:4]	RW	AddrInc	Auto address increment and packing mode on Read or Write data access. Increment occurs in word steps. Does not increment if the transaction completes with an error response or the transaction is aborted. Auto address incrementing is not performed on accesses to banked data registers `0x10-0x1C`. The status of these bits is ignored in this case. `0b11` Reserved. `0b10` Reserved. `0b01` Increment. `0b00` Auto increment OFF. The reset value is `0b00`.
[3]	-	-	Reserved, SBZ.
[2:0]	RO	Size	Size of the access to perform. Fixed at `0b010`, 32 bits. The reset value is `0b010`.

APB-AP Transfer Address Register, TAR, 0x04

Purpose Holds the address of the current transfer.

Attributes See *DAP register summary* on page 3-180 for more information.

Figure 3-219 shows the bit assignments.

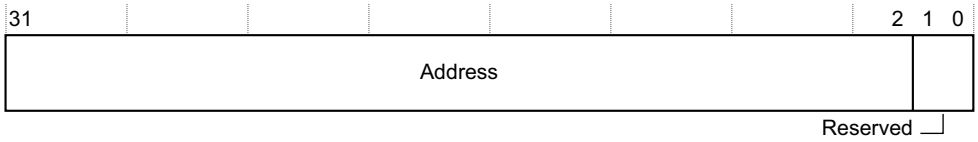


Figure 3-219 APB-AP Transfer Address Register bit assignments

Writes to the Transfer Address Register from the DAP interface write to bits [31:2] only. Bits [1:0] of **dapwdata** are ignored on writes to the Transfer Address Register.

Table 3-236 shows the bit assignments.

Table 3-236 APB-AP Transfer Address Register bit assignments

Bits	Type	Name	Function
[31:2]	RW	Address[31:2]	Address[31:2] of the current transfer. **paddr[31:2]**=TAR[31:2] for accesses from Data RW Register at `0x0C`. **paddr[31:2]**=TAR[31:4]+**dapcaddr[3:2]** for accesses from Banked Data Registers at `0x10-0x1C` and `0x0C`.
[1:0]	-	Reserved, SBZ	Set to `0b00`. SBZ/RAZ.

APB-AP Data Read/Write Register, DRW, 0x0C

Table 3-237 shows the bit assignments.

Table 3-237 ABP-AP Data Read/Write Register bit assignments

Bits	Type	Name	Function
[31:0]	RW	Data	The possible modes are: **Write mode** Data value to write for the current transfer. **Read mode** Data value read from the current transfer.

APB-AP Banked Data Registers, BD0-BD3, 0x10-0x1C

Purpose BD0-BD3 provide a mechanism for directly mapping through DAP accesses to APB transfers without having to rewrite the Transfer Address Register within a four word boundary. For example, BD0 RW from TAR, and BD1 from TAR+4.

Attributes See *DAP register summary* on page 3-180 for more information.

Table 3-238 shows the bit assignments.

Table 3-238 APB-AP Banked Data Registers bit assignments

Bits	Type	Name	Function
[31:0]	RW	Data	If **dapcaddr[7:4]** = `0x0001`, it is accessing APB-AP registers in the range `0x10-0x1C`, and the derived **paddr[31:0]** is: **Write mode** Data value to write for the current transfer to external address TAR[31:4] + **dapcaddr[3:2]** + `0b00`. **Read mode** Data value read from the current transfer from external address TAR[31:4] + **dapcaddr[3:2]** + `0b00`. Auto address incrementing is not performed on DAP accesses to BD0-BD3. The reset value is `0x00000000`.

Debug APB ROM Address, ROM, 0xF8

Purpose A ROM table must be present in all CoreSight systems.

Attributes See *DAP register summary* on page 3-180 for more information.

Figure 3-220 shows the bit assignments.

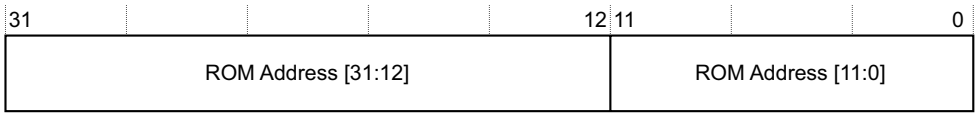


Figure 3-220 Debug APB ROM Address Register bit assignments

Table 3-239 shows the bit assignments.

Table 3-239 Debug APB ROM Address Register bit assignments

Bits	Type	Name	Function
[31:12]	RO	ROM Address [31:12]	Base address of the ROM table. The ROM provides a look-up table of all CoreSight Debug APB components. Set to `0xFFFFF` if no ROM is present. In the initial CoreSight release this must be set to `0x80000`.
[11:0]	RO	ROM Address [11:0]	Set to `0x000` if ROM is present. Set to `0xFFF` if ROM table is not present. In the initial CoreSight release this must be set to `0x000`.

APB-AP Identification Register

Figure 3-221 shows the bit assignments.

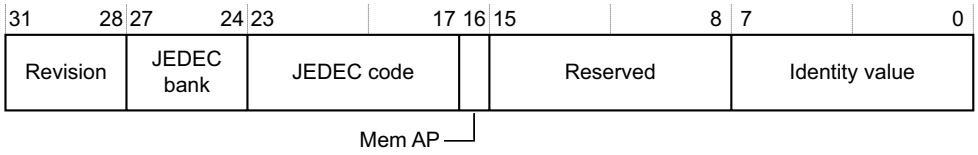


Figure 3-221 APB-AP Identification Register bit assignments

Table 3-240 shows the bit assignments.

Table 3-240 APB-AP Identification Register bit assignments

Bits	Type	Name
[31:28]	RO	Revision. Reset value is `0x4` for APB-AP.
[27:24]	RO	JEDEC bank. `0x4` indicates ARM.
[23:17]	RO	JEDEC code. `0x3B` indicates ARM.
[16]	RO	Memory AP. `0x1` indicates a standard register map is used.
[15:8]	-	Reserved, SBZ.
[7:0]	RO	Identity value. The Reset value is `0x03` for APB-AP.

3.17.5 Debug port implementation-specific registers

This section describes the implementation-specific registers.

AP Abort Register, ABORT

Purpose Present in all debug port implementations. It forces a DAP abort on SW-DP. It is also used to clear error and sticky flag conditions.

The AP Abort is always accessible, and returns an OK response if a valid transaction is received.

JTAG-DP It is at address `0x0` when the IR contains ABORT.

SW-DP It is at address `0x0` on write operations when the **APnDP** bit is equal to 0. Access to the AP Abort Register is not affected by the value of the **CTRLSEL** bit in the Select Register.

Attributes See *DAP register summary* on page 3-180 for more information

Accesses to this register always complete on the first attempt.

Figure 3-222 shows the bit assignments.

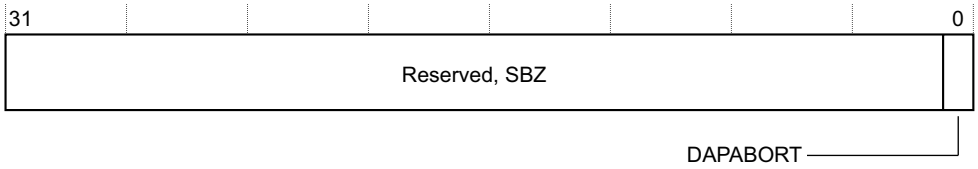


Figure 3-222 JTAG-DP AP Abort Register bit assignments

Figure 3-223 shows the bit assignments.

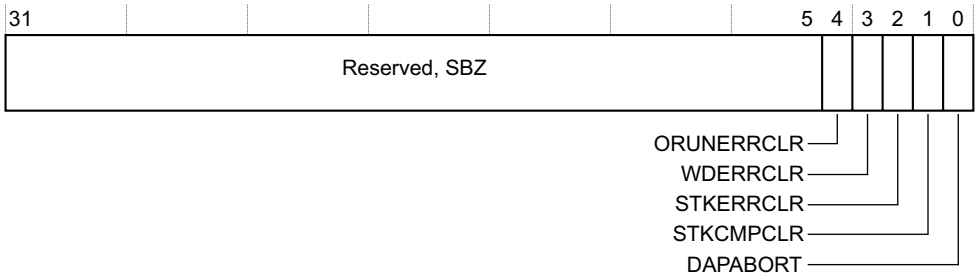


Figure 3-223 SW-DP AP Abort Register bit assignments

Table 3-241 shows the bit assignments.

Table 3-241 AP Abort Register bit assignments

Bits	Function	Description
[31:5]	-	Reserved, SBZ.
[4]	ORUNERRCLR[a]	Write 1 to this bit to clear the STICKYORUN overrun error flag[b] to 0.
[3]	WDERRCLR[a]	Write 1 to this bit to clear the WDATAERR write data error flag[b] to 0.
[2]	STKERRCLR[a]	Write 1 to this bit to clear the STICKYERR sticky error flag[b] to 0.
[1]	STKCMPCLR[a]	Write 1 to this bit to clear the STICKYCMP sticky compare flag[b] to 0.
[0]	DAPABORT	Write 1 to this bit to generate a DAP abort. This aborts the current AP transaction. Perform this only if the debugger has received WAIT responses over an extended period.

a. Implemented on SW-DP only. On a JTAG-DP this bit is Reserved, SBZ.

b. In the Control/Status Register, see *Control/Status Register, CTRL/STAT* on page 3-202.

Identification Code Register, IDCODE

Purpose Present in all debug port implementations. It provides identification information about the ARM debug Interface. The JTAG-DP is accessed using its own scan chain.

The SW-DP is at address `0b00` on read operations when the APnDP bit = 0. The value of the CTRLSEL bit in the Select Register does not affect access to the Identification Code Register. The Identification Code Register is:

- A RO Register.

- Always accessible.

Attributes See *DAP register summary* on page 3-180 for more information.

Figure 3-224 shows the bit assignments.

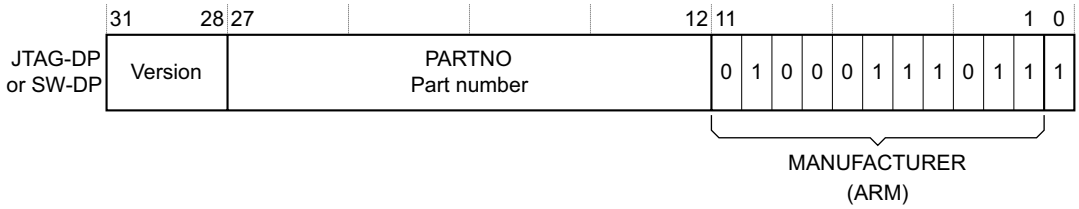


Figure 3-224 Identification Code Register bit assignments

Table 3-242 shows the bit assignments.

Table 3-242 Identification Code Register bit assignments

Bits	Function	Description
[31:28]	Version	Version code: **JTAG-DP** `0x5`. **SW-DP** `0x5`.
[27:12]	PARTNO	Part Number for the debug port. This value is provided by the designer of the debug port and must not be changed. Current ARM-designed debug ports have the following PARTNO values: **JTAG-DP** `0xBA00`. **SW-DP** `0xBA02`.
[11:1]	MANUFACTURER	JEDEC Manufacturer ID, an 11-bit JEDEC code that identifies the designer of the device. See *JEDEC Manufacturer ID*. Figure 3-224 shows the ARM value for this field as `0x23B`. This value must not be changed.
[0]	-	Always 1.

JEDEC Manufacturer ID

This code is also described as the JEP-106 manufacturer identification code, and can be subdivided into two fields, as Table 3-243 shows. JEDEC codes are assigned by the JEDEC Solid State Technology Association, see JEP106M, Standard Manufactures Identification Code.

Table 3-243 JEDEC JEP-106 manufacturer ID code, with ARM values

JEP-106 field	Bits[a]	ARM registered value
Continuation code	4 bits, [11:8]	`0b0100`, `0x4`.
Identity code	7 bits, [7:1]	`0b0111011`, `0x3B`.

a. Field width, in bits, and the corresponding bits in the Identification Code Register.

Control/Status Register, CTRL/STAT

Purpose Present in all debug port implementations. It provides control to the debug port, and status information about the debug port. JTAG-DP is at address `0x4` when the IR contains DPACC. SW-DP is at address `0b01` on read and write operations when

the APnDP bit = 0 and the CTRLSEL bit in the Select Register is set to `0b0`. For information about the CTRLSEL bit, see *AP Select Register, SELECT* on page 3-204.

The Control/Status Register is a RW register, in which some bits have different access rights. Support to some fields in the register is implementation-defined.

Attributes See *DAP register summary* on page 3-180 for more information.

Figure 3-225 shows the bit assignments.

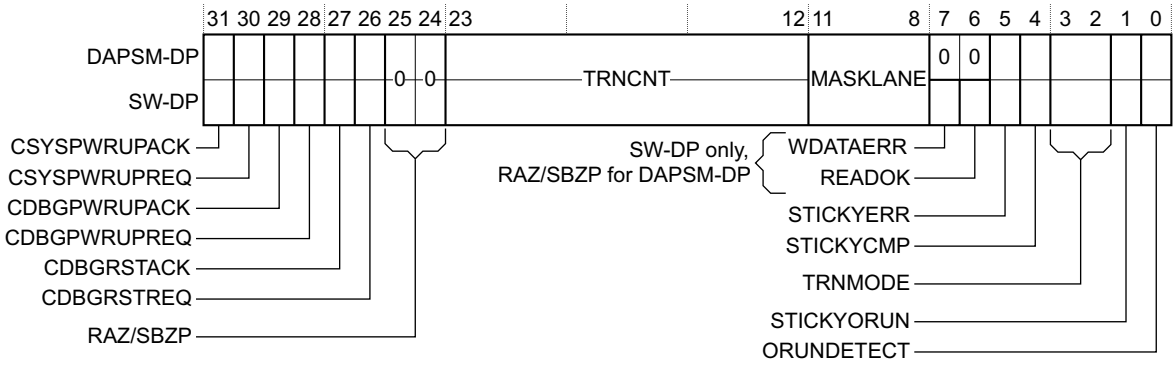


Figure 3-225 Control/Status Register bit assignments

Table 3-244 shows the Control/Status Register bit assignments.

Table 3-244 Control/Status Register bit assignments

Bits	Access	Function	Description
[31]	RO	CSYSPWRUPACK	System powerup acknowledge.
[30]	RW	CSYSPWRUPREQ	System powerup request. After a reset this bit is LOW.
[29]	RO	CDBGPWRUPACK	Debug powerup acknowledge.
[28]	RW	CDBGPWRUPREQ	Debug powerup request. After a reset this bit is LOW.
[27]	RO	CDBGRSTACK	Debug reset acknowledge.
[26]	RW	CDBGRSTREQ	Debug reset request. After a reset this bit is LOW.
[25:24]	-	-	Reserved, RAZ/SBZP.
[23:12]	RW	TRNCNT	Transaction counter. After a reset the value of this field is UNPREDICTABLE.
[11:8]	RW	MASKLANE	Indicates the bytes to be masked in pushed compare and pushed verify operations. After a reset the value of this field is UNPREDICTABLE.

Table 3-244 Control/Status Register bit assignments (continued)

Bits	Access	Function	Description
[7]	RO[a]	WDATAERR[a]	This bit is set to 1 if a Write Data Error occurs. It is set if: • there is a a parity or framing error on the data phase of a write • a write that the debug port accepted is then discarded without being submitted to the access port. This bit can only be cleared by writing `0b1` to the WDERRCLR field of the Abort Register. After a power-on reset this bit is LOW.
[6]	RO[a]	READOK[a]	This bit is set to 1 if the response to a previous access port or RDBUFF was OK. It is cleared to 0 if the response was not OK. This flag always indicates the response to the last access port read access. After a power-on reset this bit is LOW.
[5]	RO[b]	STICKYERR	This bit is set to 1 if an error is returned by an access port transaction. To clear this bit: **JTAG-DP** Write `0b1` to this bit of this register. **SW-DP** Write `0b1` to the STKERRCLR field of the Abort Register. After a power-on reset this bit is LOW.
[4]	RO[a]	STICKYCMP	This bit is set to 1 when a match occurs on a pushed compare or a pushed verify operation. To clear this bit: **JTAG-DP** Write `0b1` to this bit of this register. **SW-DP** Write `0b1` to the STKCMPCLR field of the Abort Register. After a power-on reset this bit is LOW.
[3:2]	RW	TRNMODE	This field sets the transfer mode for access port operations. After a power-on reset the value of this field is UNPREDICTABLE.
[1]	RO[a]	STICKYORUN	If overrun detection is enabled, this bit is set to 1 when an overrun occurs. To clear this bit: **JTAG-DP** Write `0b1` to this bit of this register. **SW-DP** Write `0b1` to the ORUNERRCLR field of the Abort Register. After a power-on reset this bit is LOW. See bit [0] of this register.
[0]	RW	ORUNDETECT	This bit is set to `0b1` to enable overrun detection. After a reset this bit is LOW.

a. Implemented on SW-DP only. On a JTAG-DP this bit is Reserved, RAZ/SBZP.

b. RO on SW-DP. On a JTAG-DP, this bit can be read normally, and writing `0b1` to this bit clears the bit to `0b0`.

AP Select Register, SELECT

Purpose Present in all debug port implementations. Its main purpose is to select the current access port and the active 4-word register window in that access port. On a SW-DP, it also selects the debug port address bank.

JTAG-DP It is at address `0x8` when the IR contains DPACC, and is a RW register.

SW-DP It is at address `0b10` on write operations when the APnDP bit = 0, and is a WO register. Access to the AP Select Register is not affected by the value of the CTRLSEL bit.

Attributes See *DAP register summary* on page 3-180 for more information.

Figure 3-226 on page 3-205 shows the bit assignments.

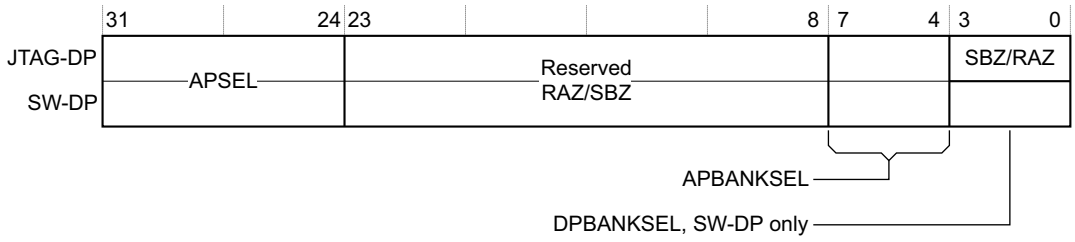


Figure 3-226 AP Select Register bit assignments

Table 3-245 shows the bit assignments.

Table 3-245 AP Select Register bit assignments

Bits	Function	Description
[31:24]	APSEL	Selects the current access port. `0x00` Selects the AP connected to master interface 0 of the DAPBUS interconnect. `0x01` Selects the AP connected to master interface 1 of the DAPBUS interconnect, if present. `0x02` Selects the AP connected to master interface 2 of the DAPBUS interconnect, if present. `0x03` Selects the AP connected to master interface 3 of the DAPBUS interconnect, if present. **...** ... **...** ... `0x1F` Selects the AP connected to master interface 31 of the DAPBUS interconnect, if present. The reset value of this field is UNPREDICTABLE.[a]
[23:8]	Reserved. SBZ/RAZ[a].	Reserved. SBZ/RAZ[a].
[7:4]	APBANKSEL	Selects the active 4-word register window on the current access port. The reset value of this field is UNPREDICTABLE.[a]
[3:0]	DPBANKSEL[b]	Selects the register that appears at DP register `0x4`. `0x0` CTRL/STAT, RW. `0x1` DLCR, RW. `0x2` TARGETID, RO. `0x3` DLPIDR, RO. All other values are reserved. Writing a reserved value to this field is UNPREDICTABLE.

a. On a SW-DP the register is write-only, therefore you cannot read the field value.
b. SW-DP only. On a JTAG-DP this bit is Reserved, SBZ/RAZ.

If **APSEL** is set to a non-existent access port, all access port transactions return zero on reads and are ignored on writes.

Note

Every ARM Debug Interface implementation must include at least one access port.

Read Buffer, RDBUFF

Purpose Present in all debug port implementations. However, there are significant differences in its implementation on JTAG and SW Debug Ports.

JTAG-DP It is at address `0xC` when the IR contains DPACC, and is a RAZ, RAZ/WI register.

SW-DP It is at address `0b11` on read operations when the APnDP bit = 0 and is a RO register. Access to the Read Buffer is not affected by the value of the CTRLSEL bit in the SELECT Register.

Attributes See *DAP register summary* on page 3-180 for more information.

Read Buffer implementation and use on a JTAG-DP

On a JTAG-DP, the read buffer always reads as zero, and writes to the read buffer address are ignored.

The read buffer is architecturally defined to provide a debug port read operation that does not have any side effects. This means that a debugger can insert a debug port read of the read buffer at the end of a sequence of operations, to return the final read result and ACK values.

Read Buffer implementation and use on a SW-DP

On a SW-DP, performing a read of the read buffer captures data from the access port, presented as the result of a previous read, without initiating a new access port transaction. This means that reading the read buffer returns the result of the last access port read access, without generating a new AP access.

After you have read the read buffer, its contents are no longer valid. The result of a second read of the Read Buffer is UNPREDICTABLE.

If you require the value from an access port register read, that read must be followed by one of:

- A second access port register read. You can read the CSW if you want to ensure that this second read has no side effects.
- A read of the DP Read Buffer.

This second access, to the access port or the debug port depending on which option you used, stalls until the result of the original access port read is available.

Wire Control Register, WCR (SW-DP only)

Purpose Present in any SW-DP implementation. Selects the operating mode of the physical serial port connection to the SW-DP.

It is a read/write register at address `0b01` on read and write operations when the CTRLSEL bit in the Select Register is set to `0b1`. For information about the CTRLSEL bit see *AP Select Register, SELECT* on page 3-204.

Note

When the CTRLSEL bit is set to `0b1`, to enable access to the WCR, the DP Control/Status Register is not accessible.

Many features of the Wire Control Register are implementation-defined.

Attributes See *DAP register summary* on page 3-180 for more information.

Figure 3-227 on page 3-207 shows the bit assignments.

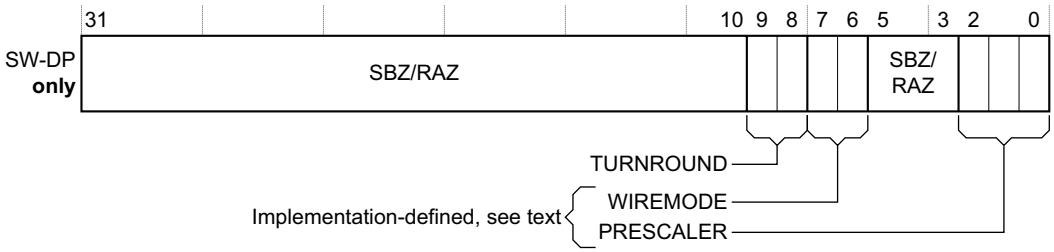


Figure 3-227 Wire Control Register bit assignments

Table 3-246 shows the bit assignments.

Table 3-246 Wire Control Register bit assignments

Bits	Function	Description
[31:10]	-	Reserved, SBZ/RAZ.
[9:8]	TURNROUND	Turnaround tristate period, see *Turnaround tristate period, TURNROUND, bits [9:8]*. After a reset this field is `0b00`.
[7:6]	WIREMODE	Identifies the operating mode for the wire connection to the debug port, see *Wire operating mode, WIREMODE, bits [7:6]* on page 3-208. After a reset this field is `0b01`.
[5:3]	-	Reserved, SBZ/RAZ.
[2:0]	PRESCALER	Reserved, SBZ/RAZ.

Turnaround tristate period, TURNROUND, bits [9:8]

This field defines the turnaround tristate period. This turnaround period permits pad delays when using a high sample clock frequency. Table 3-247 shows the permitted values of this field, and their meanings.

Table 3-247 Turnaround tristate period field bit definitions

TURNROUND[a]	Turnaround tristate period
`0b00`	1 sample period
`0b01`	2 sample periods
`0b10`	3 sample periods
`0b11`	4 sample periods

a. Bits[9:8] of the WCR.

Wire operating mode, WIREMODE, bits [7:6]

This field identifies SW-DP as operating in Synchronous mode only. This field is required, and Table 3-248 shows the permitted values of the field, and their meanings.

Table 3-248 Wire operating mode bit definitions

WIREMODE[a]	Wire operating mode
0b00	Reserved.
0b01	Synchronous, that is, no oversampling.
0b1X	Reserved.

a. Bits[7:6] of the WCR.

Target Identification Register, TARGETID (SW-DP only)

Purpose Provides information about the target when the host is connected to a single device. It is:

- A RO register.
- Accessed by a read of DP register 0x4 when the DPBANKSEL bit in the SELECT Register is set to 0x2.

The value of this register reflects the value of the **targetid[31:0]** input.

Attributes See *DAP register summary* on page 3-180 for more information.

Figure 3-228 shows the bit assignments.

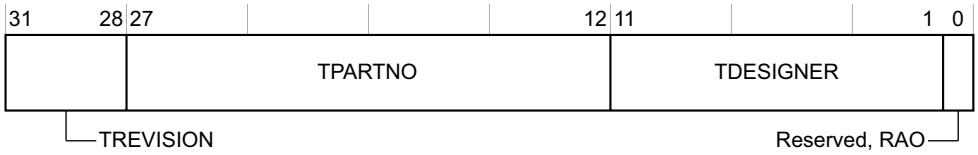


Figure 3-228 Target Identification Register bit assignments

Table 3-249 shows the bit assignments.

Table 3-249 Target Identification Register bit assignments

Bits	Function	Description
[31:28]	TREVISION	Target revision.
[27:12]	TPARTNO	IMPLEMENTATION DEFINED. This value is assigned by the designer of the part and must be unique to that part.
[11:1]	TDESIGNER	IMPLEMENTATION DEFINED. This field identifies the designer of the part. The value is based on the code assigned to the designer by JEDEC standard JEP-106, as used in IEEE 1149.1.
[0]	-	Reserved, RAO.

Data Link Protocol Identification Register, DLPIDR (SW-DP only)

Purpose Provides information about the Serial Wire protocol version. It is:

- A RO register.

- Accessed by a read of DP register 0x4 when the **DPBANKSEL** bit in the SELECT Register is set to 0x3.

The contents of this register are data link defined.

Attributes See *DAP register summary* on page 3-180 for more information.

Figure 3-229 shows the bit assignments.

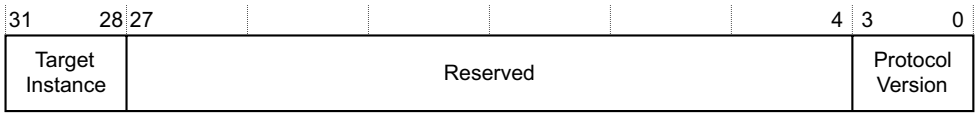


Figure 3-229 Data Link Protocol Identification Register bit assignments

Table 3-250 shows the bit assignments.

Table 3-250 Data Link Protocol Identification Register bit assignments

Bits	Function	Description
[31:28]	Target Instance	IMPLEMENTATION DEFINED. This field defines a unique instance number for this device within the system. This value must be unique for all devices that are connected together in a multi-drop system with identical values in the TREVISION fields in the TARGETID Register. The value of this field reflects the value of the **instanceid[3:0]** input.
[27:4]	-	Reserved.
[3:0]	Protocol Version	Defines the serial wire protocol version. This value is 0x1 that indicates SW protocol version 2.

Read Resend Register, RESEND (SW-DP only)

Purpose Present in any SW-DP implementation. It enables read data recovery from a corrupted debugger transfer, without repeating the original AP transfer.

It is a 32-bit read-only register at address 0b10 on read operations. Access to the Read Resend Register is not affected by the value of the DPBANKSEL bit in the SELECT Register.

Performing a read to the RESEND register does not capture new data from the access port. It returns the value that was returned by the last AP read or DP RDBUFF read.

Reading the RESEND register enables read data recovery from a corrupted transfer without having to re-issue the original read request or generate a new DAP or system level access.

The RESEND register can be accessed multiple times. It always returns the same value until a new access is made to the DP RDBUFF register or to an access port register.

Attributes See *DAP register summary* on page 3-180 for more information.

JTAG-DP register descriptions

For more information about JTAG-DP registers, their features, and how to access them, see *ARM® Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*. Also see *Common debug port features and registers* on page 4-19.

3.18 Timestamp generator register summary

Table 3-251 shows the timestamp generator registers in offset order from the base memory address.

Table 3-251 Timestamp generator register summary

Offset	Name	Type	Description
PSELCTRL region			
0x000	CNTCR	RW	*Counter Control Register, CNTCR* on page 3-212
0x004	CNTSR	RO	*Counter Status Register, CNTSR* on page 3-212
0x008	CNTCVLW	RW	Current value of counter[31:0], CNTCVLW
0x00C	CNTCVUP	RW	Current value of counter[63:32], CNTCVUP
0x020	CNTFID0	RW	Base frequency ID, CNTFID0
PSELCTRL region Management registers			
0xFD0	PERIPID4	RO	*Peripheral ID4 Register* on page 3-213
0xFD4	PERIPID5	RO	*Peripheral ID5-7 registers* on page 3-214
0xFD8	PERIPID6	RO	
0xFDC	PERIPID7	RO	
0xFE0	PERIPID0	RO	*Peripheral ID0 Register* on page 3-215
0xFE4	PERIPID1	RO	*Peripheral ID1 Register* on page 3-215
0xFE8	PERIPID2	RO	*Peripheral ID2 Register* on page 3-216
0xFEC	PERIPID3	RO	*Peripheral ID3 Register* on page 3-217
0xFF0	CIDR0	RO	*Component ID0 Register* on page 3-217
0xFF4	CIDR1	RO	*Component ID1 Register* on page 3-218
0xFF8	CIDR2	RO	*Component ID2 Register* on page 3-218
0xFFC	CIDR3	RO	*Component ID3 Register* on page 3-219
PSELREAD region			
0x000	CNTCVLW	RO	Current value of counter[31:0], CNTCVLW
0x004	CNTCVUP	RO	Current value of counter[63:32], CNTCVUP
PSELREAD region Management registers[a]			
0xFD0	PERIPID4	RO	*Peripheral ID4 Register* on page 3-213
0xFD4	PERIPID5	RO	*Peripheral ID5-7 registers* on page 3-214
0xFD8	PERIPID6	RO	
0xFDC	PERIPID7	RO	
0xFE0	PERIPID0	RO	*Peripheral ID0 Register* on page 3-215
0xFE4	PERIPID1	RO	*Peripheral ID1 Register* on page 3-215
0xFE8	PERIPID2	RO	*Peripheral ID2 Register* on page 3-216

Table 3-251 Timestamp generator register summary (continued)

Offset	Name	Type	Description
0xFEC	PERIPID3	RO	*Peripheral ID3 Register* on page 3-217
0xEE0	CIDR0	RO	*Component ID0 Register* on page 3-217
0xEE4	CIDR1	RO	*Component ID1 Register* on page 3-218
0xEE8	CIDR2	RO	*Component ID2 Register* on page 3-218
0xEEC	CIDR3	RO	*Component ID3 Register* on page 3-219

a. These are mirror registers.

3.19 Timestamp generator register description

This section describes the timestamp generator registers. Table 3-251 on page 3-210 provides cross references to individual registers.

3.19.1 Counter Control Register, CNTCR

The Counter Control Register, CNTCR, characteristics are:

Purpose Controls the counter increments.

Usage constraints There are no usage constraints.

Configurations This register is available in only in PSELCTRL configuration.

Attributes See the register summary in Table 3-251 on page 3-210.

Figure 3-230 shows the bit assignments.

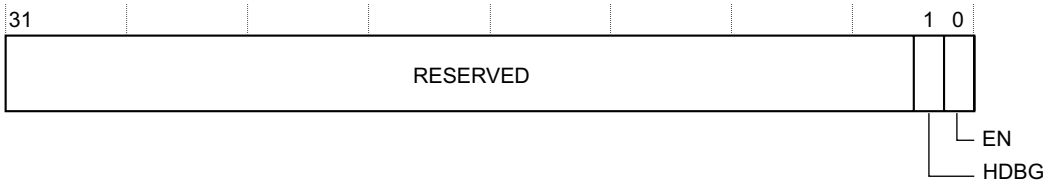


Figure 3-230 CNTCR Register bit assignments

Table 3-252 shows the bit assignments.

Table 3-252 CNTCR Register bit assignments

Bits	Name	Function	
[31:2]	UNK/SBZP	Reserved	
[1]	HDBG	Halt on Debug.	
		0	Halt on debug, **HLTDBG** signal into the counter has no effect.
		1	Halt on debug, **HLTDBG** signal into the counter halts the counter.
[0]	EN	Enable.	
		0	The counter is disabled and not incrementing.
		1	The counter is enabled and is incrementing.

3.19.2 Counter Status Register, CNTSR

The Counter Status Register, CNTSR, characteristics are:

Purpose Identifies the status of the counter.

Usage constraints There are no usage constraints.

Configurations This register is available in only in PSELCTRL configuration.

Attributes See the register summary in Table 3-251 on page 3-210.

Figure 3-231 on page 3-213 shows the bit assignments.

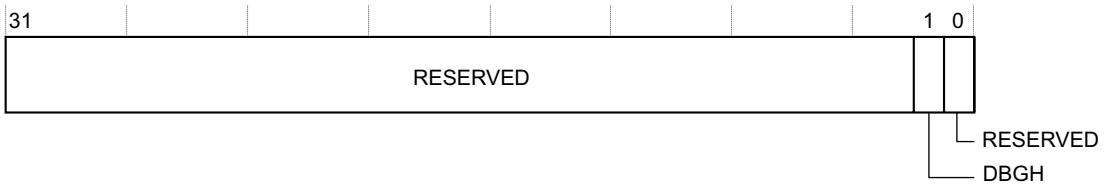


Figure 3-231 CNTSR Register bit assignments

Table 3-253 shows the bit assignments.

Table 3-253 CNTSR Register bit assignments

Bits	Name	Function
[31:2]	UNK/SBZP	Reserved.
[1]	DBGH	Debug Halted.
[0]	UNK/SBZP	Reserved.

3.19.3 CNTFID0 Register

The Counter Base Frequency ID Register, CNTFID0, characteristics are:

Purpose You must program this register to match the clock frequency of the timestamp generator, in ticks per second. For example, for a 50 MHz clock, program `0x02FAF080`.

Usage constraints There are no usage constraints.

Configurations This register is available in only in PSELCTRL configuration.

Attributes See the register summary in Table 3-251 on page 3-210.

Figure 3-232 shows the bit assignments.

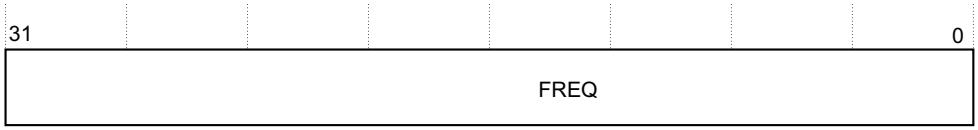


Figure 3-232 CNTFID0 Register bit assignments

Table 3-254 shows the bit assignments.

Table 3-254 CNTFID0 Register bit assignments

Bits	Name	Function
[31:0]	FREQ	Frequency in number of ticks per second. You can specify up to 4GHz.

3.19.4 Peripheral ID4 Register

The PIDR4 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-251 on page 3-210.

Figure 3-223 on page 3-201 shows the bit assignments.

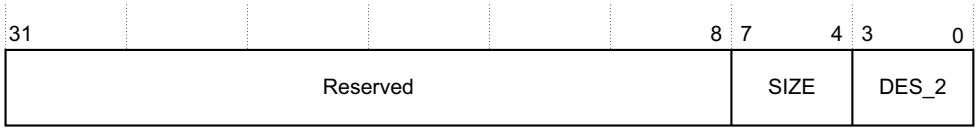


Figure 3-233 PIDR4 bit assignments

Table 3-255 shows the bit assignments.

Table 3-255 PIDR4 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	SIZE	This is a 4-bit value that indicates the total contiguous size of the memory window this component uses in powers of 2 from the standard 4KB. `0b0000` Indicates that the device only occupies 4KB of memory.
[3:0]	DES_2	This is the JEDEC JEP106 continuation code. This code, along with bits[6:4] of the identity code defined in the PIDR0 and bits[3:0] of the identity code defined in the PIDR1, identifies the designer of the component. `0b0100` JEDEC continuation code.

3.19.5 Peripheral ID5-7 registers

The PIDR5-7 characteristics are:

Purpose Reserved.

Usage constraints There are no usage constraints.

Configurations These registers are available in all configurations.

Attributes See the register summary in Table 3-251 on page 3-210.

Figure 3-234 shows the bit assignments.

Figure 3-234 PIDR5-7 bit assignments

Table 3-256 shows the bit assignments.

Table 3-256 PIDR5-7 bit assignments

Bits	Name	Function
[31:0]	Reserved	-

3.19.6 Peripheral ID0 Register

The PIDR0 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 See the register summary in Table 3-251 on page 3-210.

Figure 3-235 shows the bit assignments.

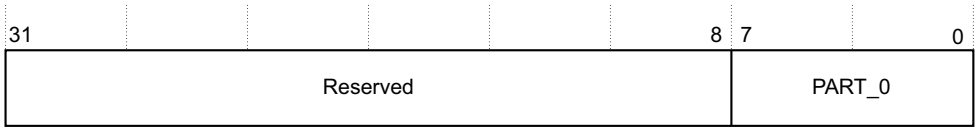


Figure 3-235 PIDR0 bit assignments

Table 3-257 shows the bit assignments.

Table 3-257 PIDR0 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PART_0	Bits[7:0] of the 12-bit part number of the component. The designer of the component assigns this part number. `0x01` Indicates bits[7:0] of the part number of the component.

3.19.7 Peripheral ID1 Register

The PIDR1 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 See the register summary in Table 3-251 on page 3-210.

Figure 3-236 shows the bit assignments.

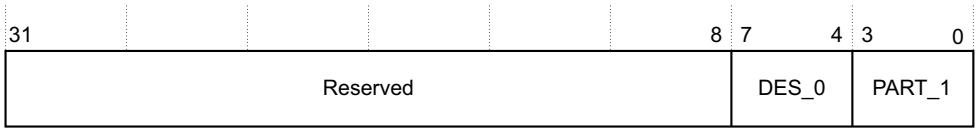


Figure 3-236 PIDR1 bit assignments

Table 3-258 shows the bit assignments.

Table 3-258 PIDR1 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	DES_0	Bits[3:0] of the JEDEC JEP106 identity code. This code, along with bits[6:4] of the identity code defined in the PIDR2 Register and the continuation code defined in the PIDR4 Register, identifies the designer of the component. `0b1011` Bits[3:0] of the JEDEC JEP106 Identity Code. The default value is ARM.
[3:0]	PART_1	Bits[11:8] of the 12-bit part number of the component. The designer of the component assigns this part number. `0b0001` Indicates bits[11:8] of the part number of the component.

3.19.8 Peripheral ID2 Register

The PIDR2 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 See the register summary in Table 3-251 on page 3-210.

Figure 3-237 shows the bit assignments.

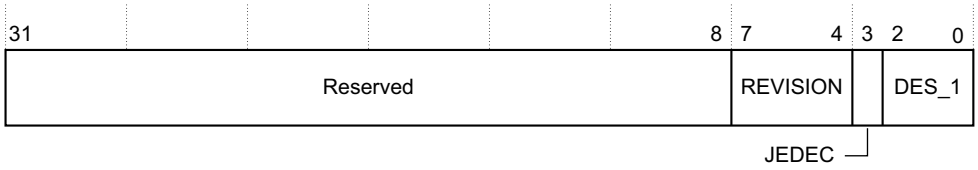


Figure 3-237 PIDR2 bit assignments

Table 3-259 shows the bit assignments.

Table 3-259 PIDR2 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	REVISION	An incremental value starting from `0b0000` for the first revision of this component. This increases by 1 for both major and minor revisions and is used to identify the major or minor revisions. `0b0000` This device is at r0p0.
[3]	JEDEC	Always set. Indicates whether the JEDEC assigned designer ID is used. `0b1` Indicates that a JEDEC assigned designer ID is used.
[2:0]	DES_1	Bits[6:4] of the JEDEC JEP106 identity code. This code, along with bits[3:0] of the identity code defined in the PIDR1 Register and the continuation code defined in the PIDR4 Register, identifies the designer of the component. `0b011` Bits[6:4] of the JEDEC JEP106 Identity Code. The default value is ARM.

3.19.9 Peripheral ID3 Register

The PIDR3 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-251 on page 3-210.

Figure 3-238 shows the bit assignments.

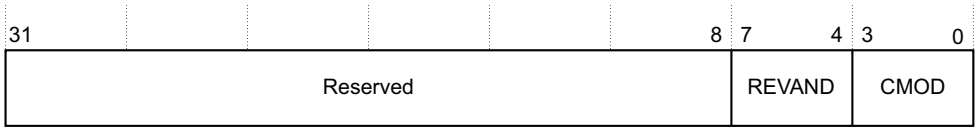


Figure 3-238 PIDR3 bit assignments

Table 3-260 shows the bit assignments.

Table 3-260 PIDR3 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	REVAND	Indicates minor errata fixes specific to the revision of the component being used, for example metal fixes after implementation. In most cases, this field is zero. ARM recommends that the component designers ensure that a metal fix can change this field if required, for example, by driving it from registers that reset to zero. `0b0000` Indicates that there are no errata fixes to this component.
[3:0]	CMOD	Indicates whether the customer has modified the behavior of the component. In most cases, this field is zero. The customer changes this value on modifications to this component. `0b0000` Indicates that the customer has not modified this component.

3.19.10 Component ID0 Register

The CIDR0 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-251 on page 3-210.

Figure 3-239 shows the bit assignments.

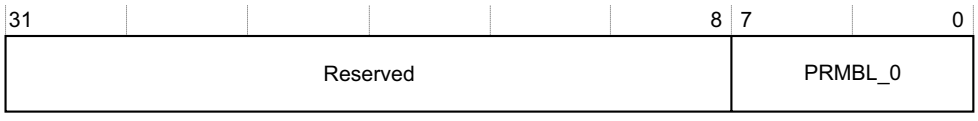


Figure 3-239 CIDR0 bit assignments

Table 3-261 shows the bit assignments.

Table 3-261 CIDR0 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_0	Contains bits[7:0] of the component identification code. `0x0D` Bits[7:0] of the identification code.

3.19.11 Component ID1 Register

The CIDR1 characteristics are:

Purpose A component identification register that indicates 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 See the register summary in Table 3-251 on page 3-210.

Figure 3-240 shows the bit assignments.

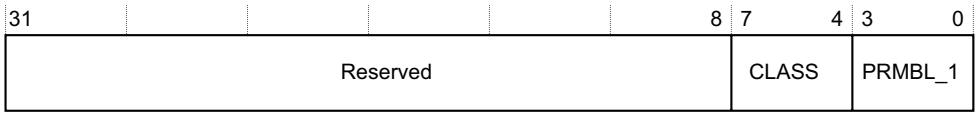


Figure 3-240 CIDR1 bit assignments

Table 3-262 shows the bit assignments.

Table 3-262 CIDR1 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:4]	CLASS	Class of the component, for example, if the component is a ROM table or a generic CoreSight component. Contains bits[15:12] of the component identification code. `0b1111` Indicates the component is a PrimeCell component.
[3:0]	PRMBL_1	Contains bits[11:8] of the component identification code. `0b0000` Bits[11:8] of the identification code.

3.19.12 Component ID2 Register

The CIDR2 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-251 on page 3-210.

Figure 3-241 shows the bit assignments.

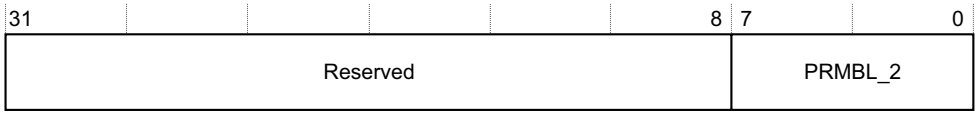


Figure 3-241 CIDR2 bit assignments

Table 3-263 shows the bit assignments.

Table 3-263 CIDR2 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_2	Contains bits[23:16] of the component identification code. `0x05` Bits[23:16] of the identification code.

3.19.13 Component ID3 Register

The CIDR3 characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes See the register summary in Table 3-251 on page 3-210.

Figure 3-242 shows the bit assignments.

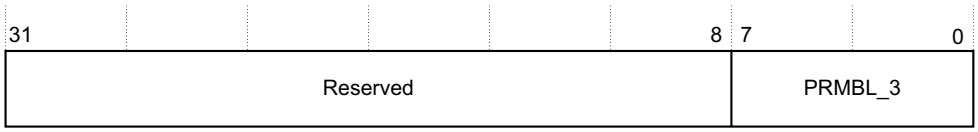


Figure 3-242 CIDR3 bit assignments

Table 3-264 shows the bit assignments.

Table 3-264 CIDR3 bit assignments

Bits	Name	Function
[31:8]	Reserved	-
[7:0]	PRMBL_3	Contains bits[31:24] of the component identification code. `0xB1` Bits[31:24] of the identification code.

Chapter 4
Debug Access Port

This chapter describes the *Debug Access Port* (DAP). It contains the following sections:

4.1 About the Debug Access Port

The DAP provides multiple master driving ports, all accessible and controlled through a single external interface port to provide system-wide debug.

The DAP is an implementation of an *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*. The DAP consists of a number of components supplied in a single configuration. All DAP components fit into the following architectural component types:

- *Debug Ports* (DPs) that are used to access the DAP from an external debugger.
- *Access Ports* (APs), to access on-chip system resources.

The debug port and access ports together are referred to as the DAP.

The DAP provides real-time access for the debugger without halting the processor to:

- AMBA system memory and peripheral registers.
- All debug configuration registers.

The DAP also provides debugger access to JTAG scan chains of system components, for example, non-CoreSight compliant processors. Figure 4-1 shows the top-level view of the functional blocks of the DAP.

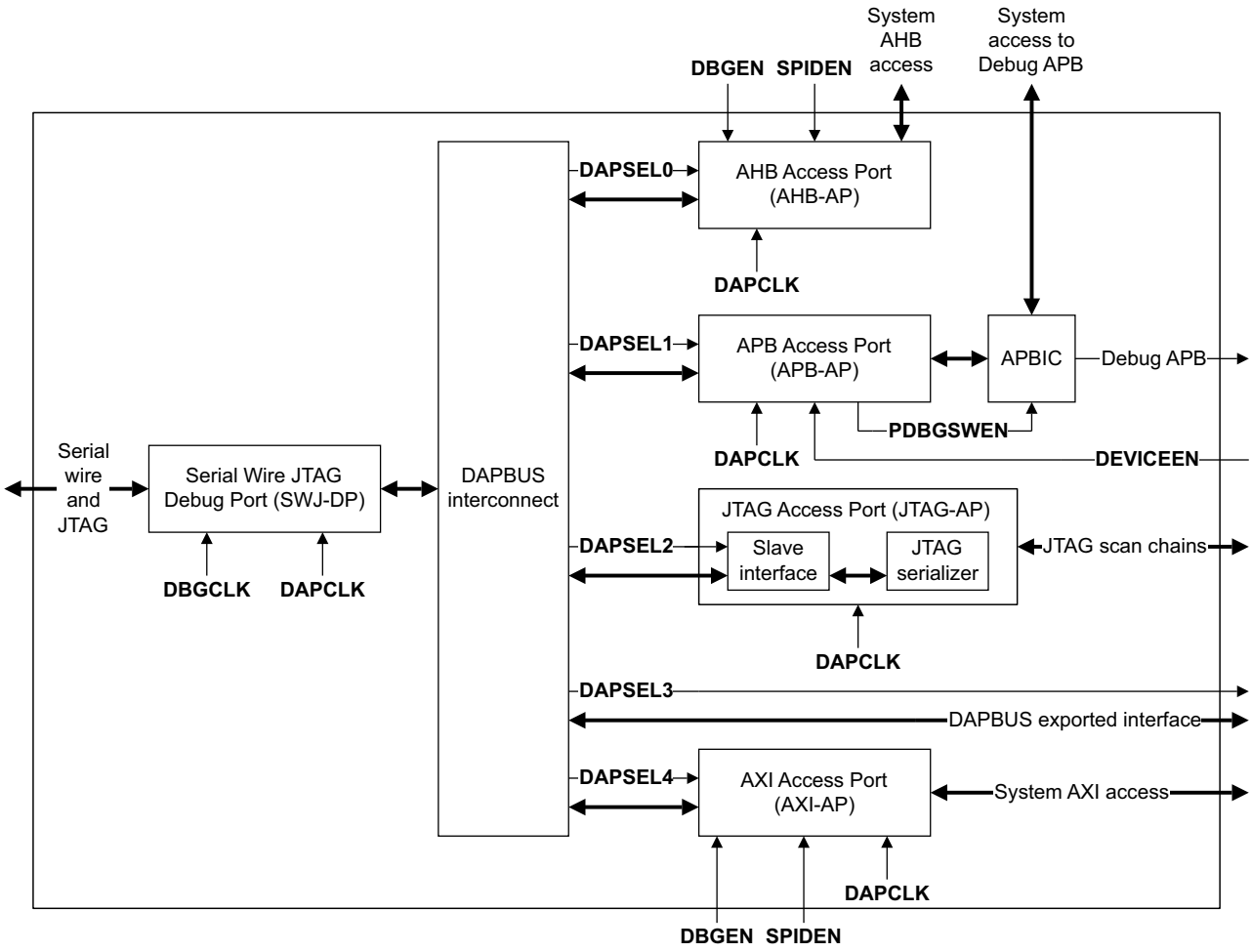


Figure 4-1 Structure of the CoreSight DAP components

Figure 4-2 on page 4-3 shows the structure of the SWJ-DP.

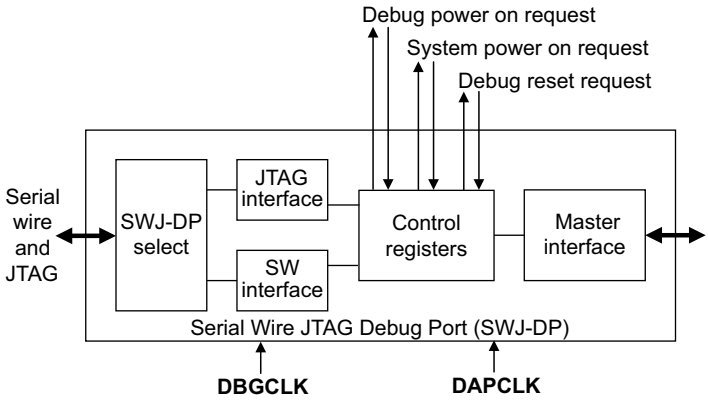


Figure 4-2 SWJ-DP

Figure 4-3 shows the structure of the AXI-AP.

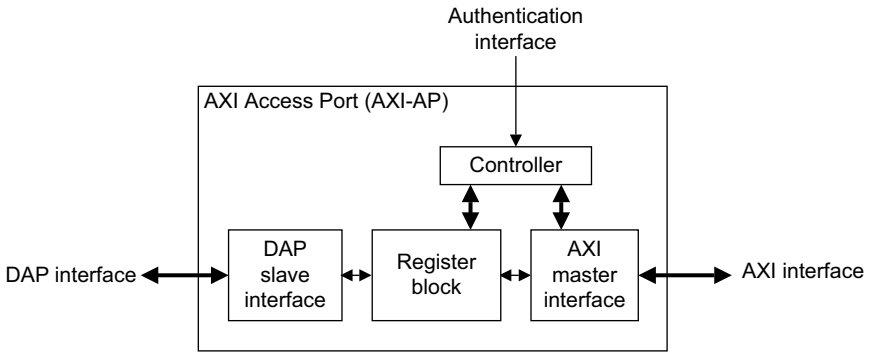


Figure 4-3 AXI-AP

Figure 4-4 shows the structure of the AHB-AP.

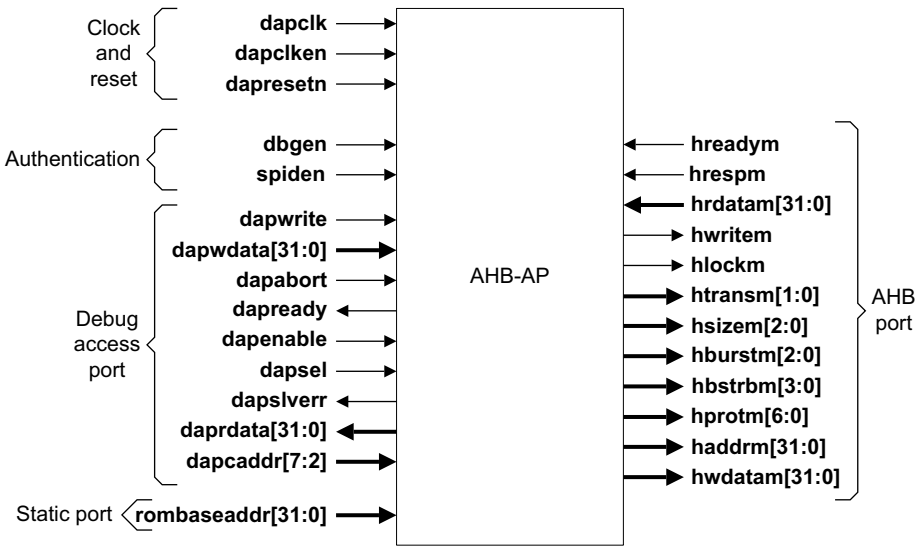


Figure 4-4 AHB-AP

Figure 4-5 on page 4-4 shows the structure of the APB-AP.

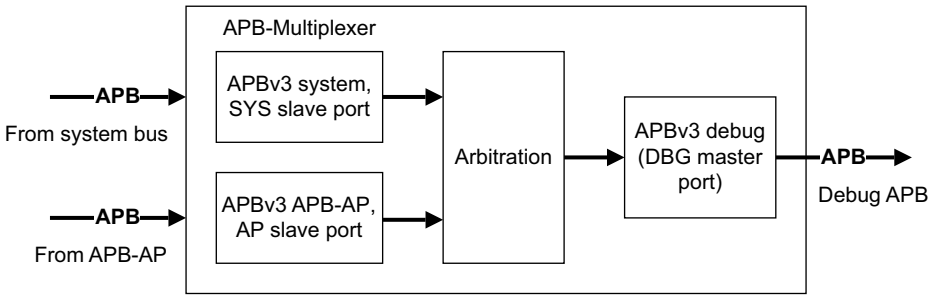


Figure 4-5 APB Access Port

The DAP enables debug access to the complete SoC using a number of master ports. Access to the CoreSight debug APB is enabled through the APB-AP and APBIC, and system access through the AHB-AP.

The DAP consists of the following interface blocks:

- External debug access using the SWJ-DP. The SWJ-DP enables selection of:
  - External serial wire access using the *Serial Wire Debug Port* (SW-DP).
  - External JTAG access using the *JTAG Debug Port* (JTAG-DP).
  - A dormant state that disables the serial wire interface to enable the connection to be shared with other protocols.
- System access using:
  - AXI-AP.
  - AHB-AP.
  - JTAG-AP.
  - DAPBUS exported interface.
  - APB-AP.
- An APB interconnect to enable system access to the CoreSight SoC components connected to the Debug APB.
- The ROM table provides a list of memory locations of the CoreSight SoC components that are connected to the Debug APB. The ROM Table is embedded within the APB interconnect. This is visible from both tools and on-chip self-hosted access. The ROM table indicates the position of all CoreSight SoC components in a system and assists in topology detection. See the *CoreSight Architecture Specification*. for more information on topology detection. For more information about the ROM Table, see Chapter 3 *Programmers Model*.

CoreSight SoC has a single multi-function DPs as follows:

SWJ-DP This is a combined debug port that can communicate in either JTAG or Serial Wire protocols as ADIv5.1 defines. It contains two debug ports, the SW-DP and the JTAG-DP, that you can select through an interface sequence to move between debug port interfaces.

The JTAG-DP is compliant with DP architecture version 0. The SW-DP is compliant with DP architecture version 2 and Serial Wire protocol version 2, that enables a SW-DP to share a target connection with other SW-DPs or other components implementing different protocols.

The access ports specified for CoreSight SoC are:

AXI-AP The AXI-AP implements the MEM-AP architecture to directly connect to an AXI memory system. You can connect it to other memory systems using a suitable bridging component.

AHB-AP The AHB-AP provides an AHB-Lite master for access to a system AHB bus. This is compliant with the *Memory Access Port* (MEM-AP) in ADIv5.1 and can perform 8 to 32-bit accesses.

JTAG-AP The JTAG-AP provides JTAG access to on-chip components, operating as a JTAG master port to drive JTAG chains throughout the ASIC. This is an implementation of the JTAG-AP in ADIv5.1.

APB-AP The APB-AP provides an APB master in AMBA v3.0 for access to the Debug APB bus. This is compliant with the MEM-AP with a fixed transfer size of 32-bit.

The DAP also implements a DAPBUS interface to enable an additional access port to be connected externally for connection to certain processors. See *DAPBUS interconnect interfaces* on page 4-16.

4.1.1 DAP flow of control

Figure 4-6 on page 4-6 shows the flow of control for the DAP when used with an off-chip debugging unit such as RealView ICE.

The DAP acts as a component to translate data transfers from one external interface format, to another internal interface. The external interface is either JTAG or serial wire. This provides a link for an external debug tool to generate accesses into an SoC. The debug port controls the JTAG-AP, AXI-AP, AHB-AP, and APB-AP through a standard bus interface:

- The JTAG-AP receives these bus transactions and translates them into JTAG instructions for control of any connected TAP controllers such as a processor.
- The AXI-AP implements the MEM-AP architecture to directly connect to an AXI memory system. You can connect it to other memory systems using a suitable bridging component.
- The AHB-AP is a bus master, along with any connected cores, on the system interconnect that can access slaves connected to that bus, for example shared memory.
- The APB-AP can only access the Debug APB. Use this to control and access CoreSight components. Control of non-debug APB peripherals is possible through the system interconnect and APBIC.

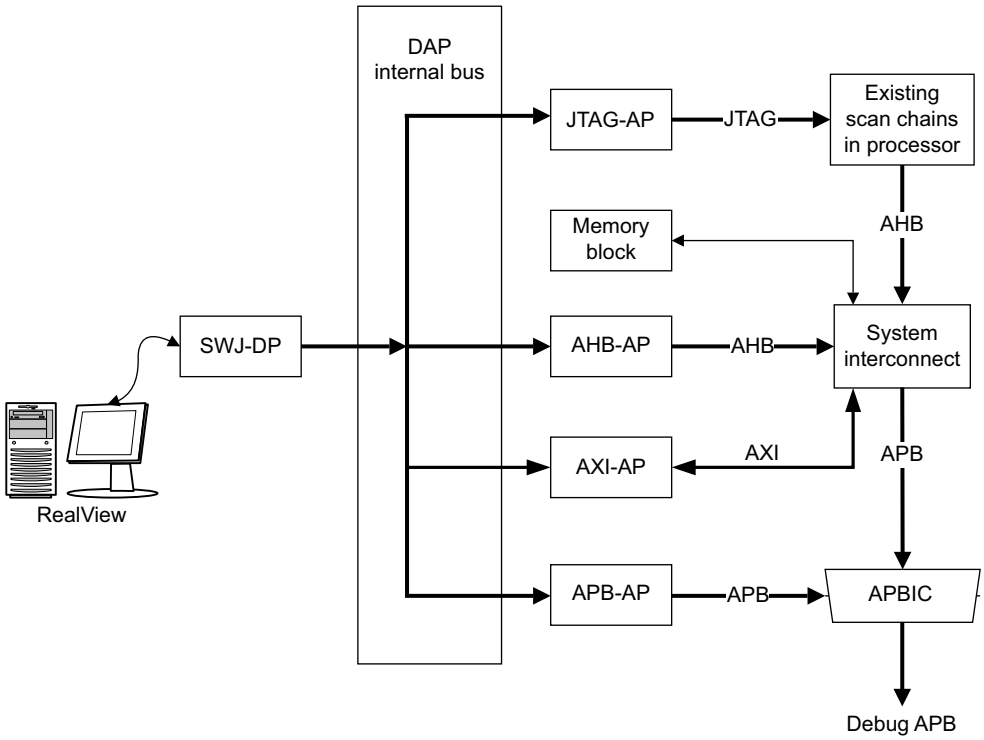


Figure 4-6 DAP flow of control

The external hardware tools, for example RealView, directly communicate with the SWJ-DP in the DAP and perform a series of operations to the debug port. Some of these accesses result in operations being performed on the DAP internal bus.

The DAP internal bus implements memory-mapped accesses to the components that are connected using the parallel address buses for read and write data. The debug port, SWJ-DP, is the bus master that initiates transactions on the DAP internal bus in response to some of the transactions that are received over the debug interface. Debug interface transfers are memory-mapped to registers in the DAP, and both the bus master and the slaves contain registers. This DAP memory map is independent of the memory maps that exist in the target system.

Some of the registers in the access ports can translate interactions into transfers on the interconnects to which they are connected. For example, in the JTAG-AP, a number of registers are allocated for reading and writing commands that result in *Test Access Port* (TAP) instructions on connected devices, for example, processors. The processor is also a bus master on a system memory structure to which the AHB-AP has access, so both the processor and AHB-AP have access to shared memory devices, or other bus slave components.

4.2 SWJ-DP

The SWJ-DP is a combined JTAG-DP and SW-DP that enables you to connect either a SWD or JTAG probe to a target. It is the standard CoreSight debug port, and enables access either to the JTAG-DP or SW-DP blocks. To make efficient use of package pins, serial wire shares, or overlays, the JTAG pins use an auto-detect mechanism that switches between JTAG-DP and SW-DP depending on which probe is connected. A special sequence on the **swdiotms** pin is used to switch between JTAG-DP and SW-DP. When the switching sequence is transmitted to the SWJ-DP, it behaves as a dedicated JTAG-DP or SW-DP depending on which sequence is performed.

Note

For more information about the programming capabilities and features of the SWJ-DP, see *Operation in JTAG-DP mode* on page 4-8 and *Operation in SW-DP mode* on page 4-9.

The following sections describe the SWJ-DP:

- *Structure of the SWJ-DP*.
- *Operation of the SWJ-DP mode*.
- *JTAG and SWD interface* on page 4-8.
- *Clock, reset, and power domain support* on page 4-14.
- *SWD and JTAG selection mechanism* on page 4-14.

4.2.1 Structure of the SWJ-DP

The SWJ-DP consists of a wrapper around the JTAG-DP and SW-DP. It selects JTAG or SWD as the connection mechanism and enables either JTAG-DP or SW-DP as the interface to the DAP.

4.2.2 Operation of the SWJ-DP mode

SWJ-DP enables you to design an *Application Specific Integrated Circuit* (ASIC) that you can use in systems that require either a JTAG interface or an SWD interface. There is a trade-off between the number of pins used and compatibility with existing hardware and test devices. There are several scenarios where you must use a JTAG debug interface. These enable:

- Inclusion in an existing scan chain, usually on-chip TAPs used for test or other purposes.
- The device to be cascaded with legacy devices that use JTAG for debug.
- Use of existing debug hardware with the corresponding test TAPs, for example in *Automatic Test Equipment* (ATE).

You can connect an ASIC that has the SWJ-DP support to legacy JTAG device without making any changes. If an SWD tool is available, only two pins are required, instead of the usual four pins used for JTAG. You can therefore use the other two pins for other purposes.

You can only use these two pins if there is no conflict with their use in JTAG mode. To support use of SWJ-DP in a scan chain with other JTAG devices, the default state after reset must be to use these pins for their JTAG function. If the direction of the alternative function is compatible driven by a JTAG debug device, the transition to a shift state can be used to transition from the alternative function to JTAG mode. You cannot use the other function while the ASIC is in JTAG debug mode.

The switching scheme is arranged so that, provided there is no conflict on the **tdi** and **tdo** pins, a JTAG debugger can connect by sending a specific sequence. The connection sequence used for SWD is safe when applied to the JTAG interface, even if hot-plugged, enabling the debugger

to continually retry its access sequence. A sequence with **tms**=1 ensures that JTAG-DP, SW-DP, and the watcher circuit are in a known reset state. The pattern used to select SWD has no effect on JTAG targets. SWJ-DP is compatible with a free-running **tck**, or a gated clock supplied by external tools.

4.2.3 JTAG and SWD interface

The external JTAG interface has four mandatory pins, **tck**, **tms**, **tdi**, and **tdo**, and an optional reset, **ntrst**. JTAG-DP and SW-DP also require a separate power-on reset, **npotrst**.

The external SWD interface requires two pins:

- A bidirectional **swdio** signal.
- A clock, **swclk**, that can be input or output from the device.

The block level interface has two pins for data and an output enable that must be used to drive a bidirectional pad for the external interface, and clock and reset signals. To enable sharing of the connector for either JTAG or SWD, connections must be made external to the SWJ-DP block. In particular, **tms** must be a bidirectional pin to support the bidirectional **swdio** pin in SWD mode. When SWD mode is used, the **tdo** pin is normally re-used for *Serial Wire Output* (SWO). You can use the **tdi** pin as an alternative input function.

Note

If you require SWO functionality in JTAG mode, you must have a dedicated pin for **traceswo**.

Operation in JTAG-DP mode

When operating as a JTAG-DP this follows the JTAG-DP as defined in the *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*. It also contains an explanation of its programmers model, capabilities, and features.

The JTAG-DP contains a debug port state machine that controls the JTAG-DP mode operation, including controlling the scan chain interface that provides the external physical interface to the JTAG-DP. It is based closely on the JTAG TAP State Machine. See *IEEE Std 1149.1-2001*.

This section contains the following:

- *Overview*.
- *Implementation-specific information* on page 4-9.

Overview

The JTAG-DP IEEE 1149.1 compliant scan chains are used to read or write register information. A pair of scan chain registers is used to access the main control and access registers within the Debug Port. They are:

- DPACC, for DP accesses.
- APACC, for AP accesses. An APACC access might access a register of a debug component of the system to which the interface is connected.

The scan chain model implemented by a JTAG-DP has the concepts of capturing the current value of APACC or DPACC, and of updating APACC or DPACC with a new value. An update might cause a read or write access to a DAP register that might then cause a read or write access to a debug register of a connected debug component. The operations available on JTAG-DP are described in the *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*. The implemented registers present within the supplied JTAG-DP are described in *JTAP-DP register descriptions* on page 3-209.

Implementation-specific information

The implementation-specific information is described in *Operation in JTAG-DP mode* on page 4-8.

Physical interface

Table 4-1 shows the physical interface for JTAG-DP and the relationship to the signal references in the *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*. The JTAG-DP interface defined in the *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2* permits an optional return clock signal. However, the CSSoC JTAG-DP implementation does not include a return clock signal.

Table 4-1 JTAG-DP physical interface

Implementation signal name, JTAG-DP	ADIv5.2 signal name, JTAG-DP	Type	JTAG-DP signal description
tdi	**DBGTDI**	Input	Debug data in
tdo	**DBGTDO**	Output	Debug data out
swclktck	**TCK**	Input	Debug clock
swditms	**DBGTMS**	Input	Debug mode select
ntrst	**DBGTRSTn**	Input	Debug TAP reset

Operation in SW-DP mode

When operating as an SW-DP Interface, this implementation is taken from the *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*, and operates with a synchronous serial interface. This uses a single bidirectional data signal and a clock signal.

Overview

The SW-DP provides a low pin count, bidirectional serial connection to the DAP with a reference clock signal for synchronous operation.

Communications with the SW-DP use a 3-phase protocol:

- A host-to-target packet request.
- A target-to-host acknowledge response.
- A data transfer phase, if required. This can be target-to-host or host-to-target, depending on the request made in the first phase.

A packet request from a debugger indicates whether the required access is to a DP register, DPACC or to an AP register, APACC, and includes a 2-bit register address. For more information about the protocol, see the *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*.

Implementation-specific information

This section contains the following:

- Clocking.
- Overview of debug interface.

Clocking

The SW-DP clock, **swclktck**, can be asynchronous to the **dapclk**. **swclktck** can be stopped when the debug port is idle.

The host must continue to clock the interface for a number of cycles after the data phase of any data transfer. This ensures that the transfer can be clocked through the SW-DP. This means that after the data phase of any transfer the host must do one of the following:

- Immediately start a new SW-DP operation.
- Continue to clock the SW-DP serial interface until the host starts a new SW-DP operation.
- After clocking out the data parity bit, continue to clock the SW-DP serial interface until it has clocked out at least 8 more clock rising edges, before stopping the clock.

Overview of debug interface

This section gives an overview of the physical interface used by the SW-DP.

Line interface

The SW-DP uses a serial wire for both host and target sourced signals. The host emulator drives the protocol timing, that is, only the host emulator generates packet headers.

The SW-DP operates in synchronous mode, and requires a clock pin and a data pin.

Synchronous mode uses a clock reference signal that can be sourced from an on-chip source and exported, or provided by the host device. The host uses this clock as a reference for generation and sampling of data so that the target is not required to perform any over-sampling.

Both the target and host are capable of driving the bus HIGH and LOW, or tri-stating it. The ports must be able to tolerate short periods of contention so that it can handle loss of synchronization.

Line pull-up

Both the host and target are able to drive the line HIGH or LOW, so it is important to ensure that contention does not occur by providing undriven time slots as part of the hand-over. So that the line can be assumed to be in a known state when neither host nor target is driving the line, a 100kΩ pull-up is required at the target, but this can only be relied on to maintain the state of the wire. If the wire is tied LOW and released, the pull-up resistor eventually brings the line to the HIGH state, but this takes many bit periods.

The pull-up is intended to prevent false detection of signals when no host device is connected. It must be of a high value to reduce IDLE state current consumption from the target when the host actively pulls down the line.

Note

Whenever the line is tied LOW, this results in a small current drain from the target. If the interface is left connected over an extended period with the target in low-power mode, the host must hold the line HIGH until the interface is re-activated.

Line turn-round

To avoid contention, a turnaround period is required whenever the device driving the wire changes.

Idle and reset

Between transfers, the host must either drive the line LOW to the IDLE state, or continue immediately with the start bit of a new transfer. The host is also free to leave the line HIGH, either driven or tri-stated, after a packet. This reduces the static current drain, but if this approach is used with a free running clock, a minimum of 50 clock cycles must be used, followed by a read ID request that initiates a new reconnection sequence.

There is no explicit reset signal for the protocol. A reset is detected by either host or target when the expected protocol is not observed. It is important that both ends of the link are reset restarting the reconnection sequence that restarts use of the protocol. Resynchronization following the detection of protocol errors or after reset is achieved by providing 50 clock cycles with the line HIGH, or tri-state, followed by a read ID request.

If the SW-DP detects that it has lost synchronization, for example, if no stop bit is seen when expected, it leaves the line undriven and waits for the host to either re-try with a new header after a minimum of one cycle with the line LOW, or signals a reset by not driving the line itself. If the SW-DP detects two bad data sequences in a row, it locks out until a reset sequence of 50 clock cycles with DBGDI HIGH is seen.

If the host does not see an expected response from SW-DP, it must allow time for SW-DP to return a data payload. The host can then retry with a read to the SW-DP ID code register. If this is unsuccessful, the host must attempt a reset.

Transfer timings

This section describes the interaction between the timing of transactions on the serial wire interface, and the DAP internal bus transfers. The section describes when the target responds with a WAIT acknowledgement.

Access port accesses result in the generation of transfers on the DAP internal bus. These transfers have an address phase and a data phase. The data phase can be extended by the access if it requires extra time to process the transaction, for example, if it must perform an AHB access to the system bus to read data.

Table 4-2 shows the terms used in Figure 4-7 on page 4-12 through Figure 4-9 on page 4-12.

Table 4-2 Terms used in SW-DP timing

Term	Description
W.APACC	Write a DAP access port register.
R.APACC	Read a DAP access port register.
xxPACC	Read or write, to debug port or access port register.
WD[0]	First write packet data.
WD[-1]	Previous write packet data. A transaction that happened before this timeframe.
WD[1]	Second write packet data.
RD[0]	First read packet data.
RD[1]	Second read packet data.

Figure 4-7 shows a sequence of write transfers. It shows that a single new transfer, WD[1], can be accepted by the serial engine, while a previous write transfer, WD[0], is completing. Any subsequent transfer must be stalled until the first transfer completes.

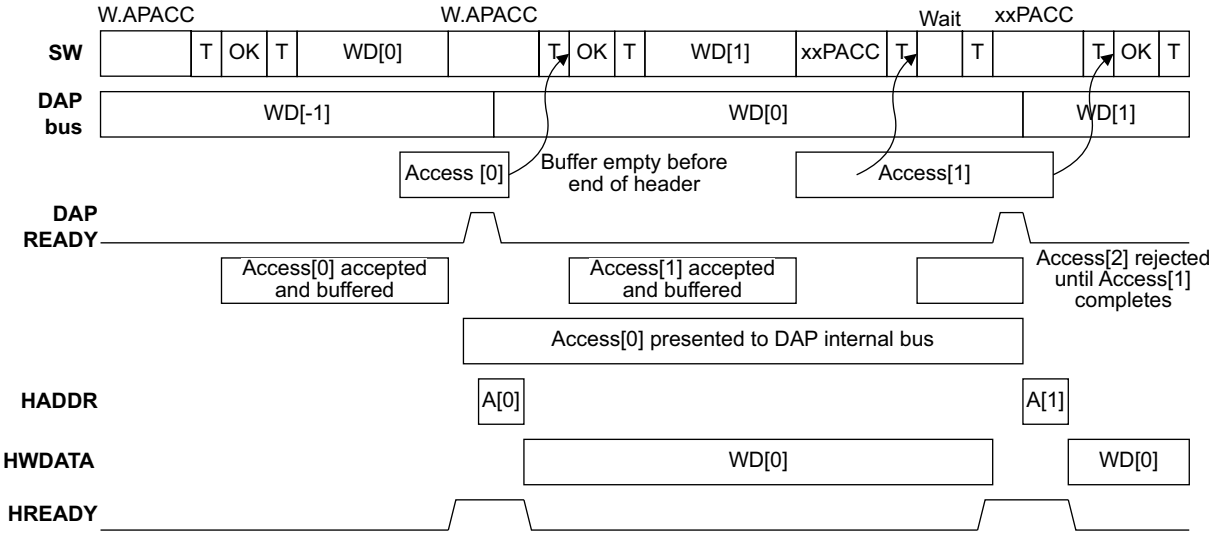


Figure 4-7 SW-DP to DAP bus timing for write

Figure 4-8 shows a sequence of read transfers. It shows that the payload for an access port read transfer provides the data for the previous read request. A read transfer only stalls if the previous transfer has not completed, in which case the first read transfer returns undefined data. It is still necessary to return data to ensure that the protocol timing remains predictable.

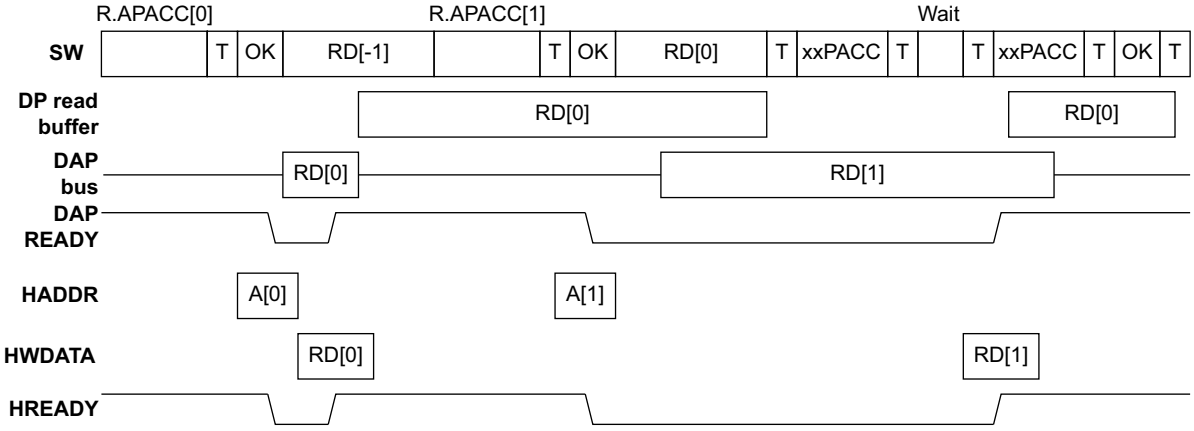


Figure 4-8 SW-DP to DAP bus timing for read

Figure 4-9 shows a sequence of transfers separated by IDLE periods. It shows that the wire is always handed back to the host after a transfer.

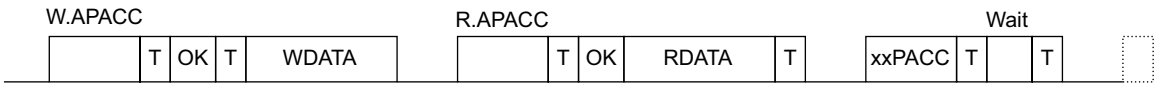


Figure 4-9 SW-DP idle timing

After the last bit in a packet, the line can be LOW, or Idle, for any period longer than a single bit, to enable the Start bit to be detected for back-to-back transactions.

SW-DP multi-drop support

The SW-DP implements the multi-drop extensions defined as part of Serial Wire protocol version 2 in the *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*. This enables multiple SW-DP implementations supporting multi-drop extensions to share a single target connection.

The multi-drop extensions are fully backwards compatible. All targets are selected following a Wire Reset, and remain selected unless a `TARGETSEL` command is received that selects a single target.

Each target must be configured with a unique combination of target ID and instance ID, to enable a debugger to select a single target to communicate with:

- The target ID is a 32-bit field that uniquely identifies the system accessed by the SW-DP.
- The instance ID is a 4-bit field that is used to distinguish between multiple instances of the same target in a system, for example, because the same chip is used more than once on a board.

The multi-drop extensions do not enable the target ID and instance ID of targets to be read when multiple targets share a connection. The debugger must either be programmed with the target ID and instance ID of each target in advance, or must iterate through a list of known target IDs and instance IDs to discover which targets are connected.

Target ID

The SW-DP target ID is configured using a 32-bit input to the SW-DP, **targetid[31:0]**. Table 4-3 shows how it must be connected.

Table 4-3 TARGETID input connections

Bits	Name	Description
[31:28]	Revision	The revision of the part. This field is not used when selecting a target.
[27:12]	Part number	Identifies the part.
[11:1]	Designer	Identifies the designer of the part. The code used is assigned by JEDEC standard JEP-106 as used in IEEE 1149.1 and CoreSight identification registers. Bits[11:8] identify the bank, and bits[7:1] identify the position within that bank.
[0]	Reserved	Must be HIGH.

The target ID must be configured even in systems where multi-drop operation is not required, because it can be used for part identification. In most cases, it can be configured with the same information as that provided in the DAP ROM table identification registers. See Chapter 3 *Programmers Model*. Table 4-4 shows the mapping of ROM table identification registers to the target ID.

Table 4-4 TARGETID mapping

TARGETID	ROM table register
[31:28]	Peripheral ID2 [7:4]
[27:24]	Peripheral ID1 [3:0]
[23:16]	Peripheral ID0 [7:0]
[15:12]	Drive LOW

Table 4-4 TARGETID mapping (continued)

TARGETID	ROM table register
[11:8]	Peripheral ID4 [3:0]
[7:5]	Peripheral ID2 [2:0]
[4:1]	Peripheral ID1 [7:4]
[0]	Drive HIGH

Instance ID

The SW-DP instance ID is configured using a 4-bit input to the SW-DP, **instanceid[3:0]**. If multiple targets with the same target ID might share a connection, **instanceid** must be driven differently for each target, for example by using non-volatile storage configured differently for each target. In most cases, you can tie this input as LOW.

4.2.4 Clock, reset, and power domain support

In the **swclktck** clock domain, there are registers to enable power control for the on-chip debug infrastructure. This enables the majority of the debug logic, for example, trace link components such as funnel, and trace sink components such as ETR, to be powered down by default. In this situation, only the serial engine must be clocked. A debug session then starts by powering up the debug sub-system. Optionally, the debugger can write to registers in the GPR, if present within the debug sub-system, to powerup debug components selectively. In SWJ-DP, either JTAG-DP or SW-DP can make powerup or reset requests but only if they are the selected device. Even in a system that does not provide a clock and reset control interface to the DAP, it is necessary to connect these signals so that it appears that a clock and reset controller is present. This permits correct handshaking of the request and acknowledge signals.

The SWJDP must be placed in an always-on domain. By instantiating an asynchronous DAPBUS bridge on the DAPBUS output of the SWJDP, the SWJDP can be power-isolated from the Debug domain.

4.2.5 SWD and JTAG selection mechanism

SWJ-DP enables one of the following modes to be selected:

- JTAG protocol.
- Serial Wire Debug protocol.
- Dormant.

When in dormant mode, the **tms**, **tdi**, and **tdo** signals can be used for other purposes, enabling other devices connected to the same pins to use alternative debug protocols.

The switcher defaults to JTAG operation on power-on reset. Therefore the JTAG protocol can be used from reset without sending a selection sequence.

The SWJ-DP contains a mode status output, **jtagnsw**, that is HIGH when the SWJ-DP is in JTAG mode and LOW when in SWD or Dormant mode. This signal can be used to:

- Disable other TAP controllers when the SWJ-DP is in SWD or dormant mode, for example by disabling **tck** or forcing **tms** HIGH.
- Multiplex the SWO, **traceswo**, onto another pin such as **tdo** when not in JTAG mode.

Another status output, **jtagtop**, indicates the state of the JTAG-DP TAP controller. The states are:

- Test-Logic-Reset.
- Run-Test/Idle.
- Select-DR-Scan.
- Select-IR-Scan.

This signal can be used with **jtagnsw** to control multiplexers so that, for example, **tdo** and **tdi** can be reused as *General Purpose Input/Output* (GPIO) signals when the device is not in JTAG mode, or during cycles when these signals are not in use by the JTAG-DP TAP controller.

See the *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2* for information on the SWJ-DP switching sequences.

4.3 DAPBUS interconnect interfaces

This section describes the DAPBUS interconnect interfaces in:

- *Clock and reset*.
- *Functional interfaces*.

4.3.1 Clock and reset

There are no clock or reset pins as this component is a combinational block.

4.3.2 Functional interfaces

The DAPBUS interconnect has one DAPBUS slave interface. It has a configurable number of DAPBUS master interfaces. This is automatically defined at design time.

4.3.3 Normal operating modes

To address a particular *Access Port* (AP), the *Debug Port* (DP) uses the eight MSBs of its address bus, **DAPCADDRS[15:2]**. The value driven on these address lines is determined by the **APSEL[7:0]** field in the AP Select register. This register is present on all DP implementations that are compliant with the *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*.

Accessing the default slave returns a **DAPREADYS** without an error on **DAPSLVERRS**.

4.4 DAP asynchronous bridge

The DAP bus asynchronous bridge enables data transfer between two asynchronous clock domains. It also provides an optional LPI to support its use between two power domains.

4.4.1 Clock and reset

The bridge operates in a single clock domain with one asynchronous reset. The clocks and resets of the DAP asynchronous bridge are:

DAPCLKM DAP master interface clock.

DAPCLKS DAP slave interface clock.

DAPRESETMN DAP master interface reset.

DAPRESETSN DAP slave interface reset.

4.4.2 Functional interface

The DAP asynchronous bridge has the following interfaces:

- One DAPBUS master interface.
- One DAPBUS slave interface.
- One optional LPI slave interface.

4.4.3 Functional description

The following are the main modes of device operation:

Normal operation

The beginning of a transfer is indicated by **DAPSELS** being asserted HIGH. The bridge transfers the control information to the master interface while maintaining **DAPREADYS** LOW. The bridge indicates the completion of transfer by driving **DAPREADYS** HIGH after it detects the transfer completion on the master interface.

Aborting the current transaction

A transaction in progress can be aborted by driving DAPABORTS to HIGH. The bridge responds to the abort by driving the DAPREADYS HIGH. However, the next transaction is stalled till the previously aborted transaction completes on the master interface.

4.4.4 Low-power support

The DAP asynchronous bridge supports an optional LPI to the power controller. The system level power controller can request the master interface of the bridge to go into low-power state through the LPI. The bridge enters low-power state when there are no pending transactions. When the bridge is in low-power mode, a wake-up request is generated on a new transaction and the transaction completes when the power controller brings the master interface out of low-power state. The bridge stalls the transaction on the slave interface until the master interface is brought out of low-power state, and ensures that there is no loss of data transferred through the bridge.

4.5 DAP synchronous bridge

The DAPBUS synchronous bridge enables data transfer between two synchronous clock domains. It also includes an optional LPI for power-gating.

4.5.1 Clock and reset

The DAP synchronous bridge operates in a single clock domain with one asynchronous reset. The clocks and resets of the DAP synchronous bridge are:

DAPCLK DAP synchronous clock.

DAPRESETn DAP synchronous reset.

4.5.2 Functional interface

The DAP synchronous bridge has the following interfaces:

- One DAPBUS master interface.
- One DAPBUS slave interface.
- One optional LPI slave interface.

4.5.3 Functional description

The DAP synchronous bridge functions in the same way as the APB synchronous bridge with the exception of the **DAPABORT** signal. See *APB synchronous bridge* on page 4-40.

Abort function

An ongoing transfer can be aborted by asserting **DAPABORTS** HIGH. The bridge acknowledges a successful abort by signalling **DAPREADYS** HIGH.

4.6 Common debug port features and registers

This section describes specific information about features and registers that are present within this implementation of SW-DP and JTAG-DP as part of the SWJ-DP. For more information about all the features and registers present within SW-DP and JTAG-DP, see *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*. This section contains the following implementation-specific information:

- *Features overview*.
- *Example pushed operations*.

4.6.1 Features overview

Both the SW-DP and JTAG-DP views within the SWJ-DP contain the same features as those that are defined in the *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*. The following features are included:

- Sticky flags and debug port error responses as a result of either a read and write error response from the system or because of an overrun detection, STICKYORUN.
- Pushed compare and pushed verify to enable more optimized control from a debugger by performing a set of write transactions and enabling any comparison operation to be done within the debug port. See *Example pushed operations*.
- Transaction counter to recover to a point within a repeated operation.
- System and debug power and debug reset control. This is to enable an external debugger to connect to a potentially turned-off system and to powerup as much as is required to get a basic level of debug access with minimal understanding of the system.

For more information, see *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*.

4.6.2 Example pushed operations

These are two examples that use this specific implementation of the *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*. All register and feature references relate to this specification.

This section contains the following examples:

- Example 4-1.
- Example 4-2 on page 4-20.

Example 4-1 Example use of pushed verify operation on an AHB-AP

You can use pushed verify to verify the contents of system memory as follows:

1. Make sure that the AHB-AP *Control/Status Word* (CSW) is set up to increment the *Transfer Address Register* (TAR) after each access.
2. Write to the TAR to indicate the start address of the Debug Register region that is to be verified.
3. Write a series of expected values as access port transactions. On each write transaction, the debug port issues an access port read access, compares the result against the value supplied in the access port write transaction, and sets the **STICKYCMP** bit in the CRL or STAT Register if the values do not match. The TAR is incremented on each transaction.

In this way, the series of values supplied is compared against the contents of the access port locations, and **STICKYCMP** is set if they do not match.

Example 4-2 Example use of pushed find operation on an AHB-AP

You can use pushed find to search system memory for a particular word. If you use pushed find with byte lane masking you can search for one or more bytes.

1. Make sure that the AHB-AP CSW is set up to increment the TAR after each access.
2. Write to the TAR to indicate the start address of the Debug Register region that is to be searched.
3. Write the value to be searched for as an AP write transaction. The debug port repeatedly reads the location indicated by the TAR. On each debug port read:
   - The value returned is compared with the value supplied in the access port write transaction. If they match, the **STICKYCMP** flag is set.
   - The TAR is incremented.

This continues until **STICKYCMP** is set, or **ABORT** is used to terminate the search.

You can also use pushed find without address incrementing to poll a single location, for example, to test for a flag being set on completion of an operation.

4.7 Access ports

An access port provides the interface between the debug port interface and one or more debug components present within the system. There are two kinds of access port supplied with this DAP:

- JTAG-AP for connecting to on-chip based debug TAPs.
- MEM-AP, designed for connection to memory bus system with address and data controls.

All access ports follow a base standard for identification, and debuggers must be able to recognize and ignore access ports that they do not support. The connection method does not depend on the type of debug port used and the type of access port being accessed.

For more information on access ports and recommend debugger interaction with access ports, see the *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*.

4.7.1 Overview

There are four access ports supplied in the DAP. It is possible to connect a fourth access port externally. The supplied access ports within this release are:

- JTAG-AP to control up to eight scan chains.
- AXI-AP for connection to an AXI interconnect.
- AHB-AP for connection to the main system bus.
- APB-AP to enable direct connection to the dedicated debug bus.

4.8 JTAG-AP

The *JTAG Access Port* (JTAG-AP) provides JTAG access to on-chip components, operating as a JTAG master port to drive JTAG chains throughout a SoC. The JTAG command protocol is byte-oriented, with a word wrapper on the read and write ports to yield acceptable performance from the 32-bit internal data bus in the DAP. Daisy chaining is avoided by using a port multiplexer. In this way, slower cores do not impede faster cores. For more information about the JTAG-AP, see the description of the JTAG-AP in the *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*.

The implementation-specific features of the JTAG-AP are described in the following sections:

- *External interfaces*.
- *RTCK connections*.

4.8.1 External interfaces

Table 4-5 shows the JTAG to slave device signals.

Each of the eight JTAG scan chains are on the same bit positions for each JTAG signal. For example, connections for scan chain 0 can be located on bit [0] of each bus connection of **cstck**, **cstms**, **cstdi**, and **portconnected**.

Table 4-5 JTAG to slave device signals

Name	Type	Description
nsrstout[7:0]	Output	Subsystem reset out.
srstconnected[7:0]	Input	Subsystem reset is present.
ncstrst[7:0]	Output	JTAG test reset.
cstck[7:0]	Output	JTAG test clock.
cstms[7:0]	Output	JTAG test mode select.
cstdi[7:0]	Output	JTAG test data in, to external TAP.
cstdo[7:0]	Input	JTAG test data out, from external TAP.
csrtck[7:0]	Input	Return test clock, target pacing signal.
portconnected[7:0]	Input	JTAG port is connected, status signal.
portenabled[7:0]	Input	JTAG port is enabled, for example, it might be deasserted by a processor powering down.

4.8.2 RTCK connections

This section describes the **rtck** connections.

Global port and RTCK

When more than one bit of **portsel[7:0]** is set, all active JTAG-AP multiplexer port **rtcks** are combinatorially joined, so that:

- If **tck**=0 then select OR of active **rtcks**.
- **tck**=1 then select AND of active **rtcks**.

An active **rtck** is generated by an active port that is defined as a port which:

- Is selected, when its **portsel[7:0]** bit is set.
- Is connected, when its **portconnected[7:0]** bit is set.

- Has not been disabled or powered down in this session, when its PSTA bit is 0.

If no ports are active, **rtck** is connected directly to **tck**. This means that disabling or powering down a JTAG slave cannot lock up the **rtck** interface.

Asynchronous TAP controllers that do not require an **rtck** connection must connect their **tck** output from JTAG-AP to the corresponding **rtck** input.

RTCK wrapper

Note

This section applies only to synchronous TAP controllers.

Where devices do not have a return clock, an **rtck** wrapper must be used to register **tck** against the processor clock. See *Additional reading* on page ix and the applicable Integration Manual for information on how to implement an **rtck** wrapper.

4.9 AXI-AP

The AXI-AP implements the MEM-AP architecture to directly connect to an AXI memory system. You can connect it to other memory systems using a suitable bridging component.

For more information on the key features and the configuration options of the AXI-AP, see *Additional reading* on page ix.

4.9.1 AXI-AP integration overview

Figure 4-10 shows an example integration of AXI-AP within CoreSight SoC DAP.

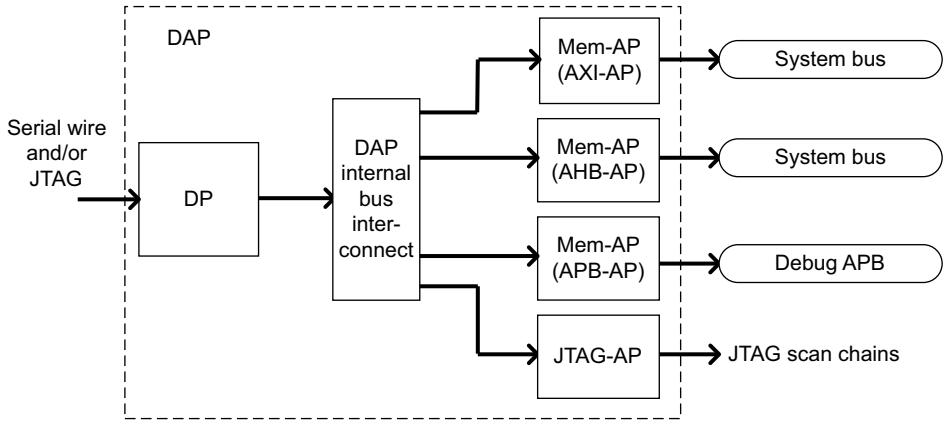


Figure 4-10 AXI-AP example integration with CoreSight SoC DAP

4.9.2 Clock and reset

The AXI-AP has a single clock domain as input, **CLK**, and a single reset, **RESETn**. The **RESETn** must only be asserted LOW when there is no pending transaction on the AXI interface.

The AXI-AP resets are asserted asynchronously but deasserted synchronously to the **CLK**.

4.9.3 Functional interfaces

AXI-AP has the following bus interfaces:

- DAP internal slave interface that connects to DP.
- Authentication slave interface.
- AXI4 master interface.

4.9.4 AXI-AP functionality

The following sections describe the AXI-AP functionality:

- *AXI-AP reset*.
- *DAP transfer abort* on page 4-25.
- *Error responses* on page 4-25.
- *Valid combinations of AxCACHE and AxDOMAIN* on page 4-29.

AXI-AP reset

On AXI-AP reset, the entire AXI-AP module is reset and the transaction history is lost.

ARM recommends that reset must not be asserted while AXI transfer is ongoing. However, AXI-AP permits reset to be asserted with the understanding that all transaction history is lost.

DAP transfer abort

On DAP transfer abort, the AXI-AP asserts **DAPREADY** one cycle after **DAPABORT**. The DAP transfer abort does not cancel the ongoing AXI transfer.

Error responses

This section describes the following:

- *AXI initiated error responses*.
- *AP initiated error response*.
- *AXI and AP initiated error responses* on page 4-26.

AXI initiated error responses

An error response received on the AXI master interface propagates onto the DAP bus as the transfer is completed.

In case of a 64-bit data transfer, a sequence of two reads or writes must be generated on the DAP bus for a single 64-bit access on the AXI interface. In case of reads, the first read request on the DAP bus sends a read request on the AXI interface while in case of writes, a write access is sent on the AXI interface only after two write requests are received on the DAP bus.

Therefore, an error response received for a read request is for the first read request on DAP bus while an error response received for a write request is for the second write request on the DAP bus.

AP initiated error response

AXI-AP writes after an abort

After a **DAPABORT** operation is carried out, and an external transfer is still pending, that is, the transfer in progress bit remains HIGH, all writes to the AXI-AP return an error response which you can ignore.

AXI-AP writes return an error until the transfer in progress, the **TrInProg** bit, is cleared when the system transfer completes.

AXI-AP reads after a 64-bit AXI read sequence is broken

Read requests from the DAP bus must access both BDx registers of the pair, and must access the lower-numbered register first. For a DRW, it must access it twice to get the entire 64-bit word from AXI interface.

All other accesses, such as a read followed by a write access to the same or different registers, return an error response by asserting **DAPSLVERR**.

AXI-AP writes after a 64-bit write sequence is broken

Write requests from DAP I/F must access both BDx registers of the pair and must access the lower-numbered register first. For a DRW, it must access twice to build a 64-bit packet as write data on the AXI interface.

All other accesses, such as a write followed by another R/W access to different registers, return an error response.

For example, after accessing DRW, the next access on the DAP bus must be a write to DRW. Any other access returns an error response.

Similarly, after accessing BD0, the next access must be a write to BD1. Any other access returns an error response.

Aborted AXI barrier transaction

It is possible to abort a barrier transaction that has not yet completed. When the abort request is generated, the **DAPREADY** is asserted HIGH in the next cycle. However the **CSW.TrInPrg** bit remains set to indicate that the AXI-Interface is busy waiting to complete the transaction. While the AXI-Interface is busy, a R/W request to DRW or BD*x* registers that results in a transaction on the AXI interface, causes the AXI-AP to return an error response by asserting **DAPSLVERR**.

AXI and AP initiated error responses

If DAPSLVERR is HIGH and **TrInProg** is LOW in the CSW Register, the error is from either:

- A system error response if **registered versions of DBGEN** and **SPIDEN** permit the transfer to be initiated.
- An AXI-AP error response if **registered versions of DBGEN** and **SPIDEN** do not permit the transfer.

Table 4-6 shows the difference between AXI and AP initiated error response.

Table 4-6 Difference between AXI and AP initiated error response

Sprot	SPIDEN_registered	DBGEN	Error response from	Reason
X	X	0	AXI-AP	All transfers blocked
0	0	1	AXI-AP	Secure transfers blocked
0	1	1	System	Secure transfer produced an error response
1	X	1	System	Non secure transfer produced an error response

If **DAPSLVERR** is HIGH and **TrInProg** is HIGH, then the error is from an access port error response. This case can only occur after the initiation of an abort when the system transfer has not completed.

AXI-AP

Table 4-7 shows the implemented features.

Table 4-7 AXI-AP features at a glance

Feature	Comment
AXI4 interface support	-
Auto-incrementing TAR	-
Stalling accesses	-
Access size	8, 16, 32, or 64 bits.
Endianness	Little-endian.
Error response	-
Packed transfers	-
ROM table pointer register	-

Table 4-7 AXI-AP features at a glance (continued)

Feature	Comment
Long address support	-
AXI transfers	Write, read transfers.
	Burst size of 1 only.
	No out-of-order transactions.
	No multiple outstanding accesses.
	Only aligned transfers are supported.
ACE-Lite	Limited set of commands to support coherency in the system.
	All transactions to non-shareable memory regions.
	Limited subset of transactions to shareable memory regions.
	For reads only. Supports the ReadOnce transaction type.
	For writes only. Supports the WriteUnique transaction type.
	Barrier transactions

AXI transfers

The AMBA4 AXI compliant Master Port supports the following features:

- Bursts of single transfer.
- Master processes one transaction at a time in the order they are issued.
- No out-of-order transactions.
- No issuing of multiple outstanding addresses.

Burst length

The AXI-AP supports burst length of one transfer only. **ARLEN[3:0]** and **AWLEN[3:0]** are always `0b0000`.

Packed 8 or 16-bit transfers are treated as individual burst lengths of one transfer at the AXI interface. This ensures that there are no issues with boundary wrapping to avoid additional AXI-AP complexity.

Burst size

Supported burst sizes are:

- 8-bit.
- 16-bit.
- 32-bit.
- 64-bit.

Burst type

ARBURST or **AWBURST** is always `0b00`.

Because only bursts of one transfer are supported, burst type has no meaning in this context.

Atomic accesses

AXI-AP supports normal accesses only.

ARLOCK[1:0] and **AWLOCK[1:0]** signals are always `0b00`.

Unaligned accesses

Unaligned accesses are not supported. Depending on the size of the transfers, addresses must be aligned.

- For 16-bit half word transfers:
  - Base address `0x01` is aligned and **AxADDR[7:0]** = `0x00`.
  - Base address `0x02` is retained and **AxADDR[7:0]** = `0x02`.
- For 32-bit word transfers:
  - Base address `0x01` to `0x03` is aligned and **AxADDR[7:0]** = `0x00`.
  - Base address `0x04` is retained and **AxADDR[7:0]** = `0x04`.
- For 64-bit word transfers:
  - Base address `0x04` is aligned and **AxADDR[7:0]** = `0x00`.
  - Base address `0x08` is retained and **AxADDR[7:0]** = `0x08`.

For example, for 16-bit transfers, addresses must be aligned to 16-bit half-word boundary, for 32-bit word transfers, addresses must be word-aligned, and for 64-bit double word transfers, addresses must be double-word aligned.

Packed Transfers

32-bit Transactions

The DAP internal interface is a 32-bit data bus. However 8-bit or 16-bit transfers can be formed on AXI according to the size field in the CSW register, `0x000`. The AddrInc field in the CSW Register permits optimized use of the DAP internal bus to reduce the number of accesses to the DAP. It indicates whether the entire data word can be used to pack more than one transfer. If packed transfers are initiated, then address incrementing is automatically enabled. Multiple transfers are carried out in sequential addresses, with the size of the address increment based on the size of the transfer.

An example of the transactions are:

For an unpacked 16-bit write at a base address of base `0x2`, that is, CSW[2:0]=`0b001`, CSW[5:4]=`0b01`, **WDATA[31:16]** is written from bits [31:16] in the DRW register.

For an unpacked 8-bit read at a base address of base `0x1`, that is, CSW[2:0]=`0b000`, CSW[5:4]=`0b01`, **RDATA[31:16]** and **RDATA[7:0]** are zero, **RDATA[15:8]** contains read data.

For a packed byte write at base address of base `0x2`, that is, CSW[2:0]=`0b000` and CSW[5:4]=`0b10`, four write transfers are initiated, and the order of data that is sent is:

- **WDATA[23:16]**, from **DRW[23:16]** to **AWADDR[31:0]**=`0x00000002`.
- **WDATA[31:24]**, from **DRW[31:24]** to **AWADDR[31:0]**=`0x00000003`.
- **WDATA[7:0]**, from **DRW[7:0]** to **AWADDR[31:0]**=`0x00000004`.
- **WDATA[15:8]**, from **DRW[15:8]** to **AWADDR[31:0]**=`0x00000005`.

For a packed half-word read at a base address of base `0x2`, that is, CSW[2:0]=`0b001`, CSW[5:4]=`0b10`, two read transfers are initiated:

- **RDATA[31:16]** is stored into **DRW[31:16]** from **ARADDR[31:0]**=`0x00000002`.
- **RDATA[15:0]** is stored into **DRW[15:0]** from **ARADDR[31:0]**=`0x00000004`.

The AXI-AP only asserts **DAPREADY** HIGH when all packed transfers from the AXI interface have completed.

If the current transfer is aborted or the current transfer receives an ERROR response, the AXI-AP does not complete the subsequent packed transfers and returns **DAPREADY** HIGH immediately after the current packed transfer.

Valid combinations of AxCACHE and AxDOMAIN

Table 4-8 shows the valid combinations of **AxCACHE** and **AxDOMAIN**.

Table 4-8 Valid combination of AxCACHE and AxDOMAIN values

AxCACHE	Access Type	AxDOMAIN	Domain Type	Valid
0000	Device	00	Non-shareable	NO
0001		01	Inner-shareable	NO
		10	Outer-shareable	NO
		11	System	YES
0010	Non-Cacheable	00	Non-shareable	Enabled
0011		01	Inner-shareable	Enabled
		10	Outer-shareable	Enabled
		11	System	YES
010x	-	-	-	NO
100x	-	-	-	
110x	-	-	-	
011x	Write	00	Non-shareable	YES
101x	Through	01	Inner-shareable	YES
111x	Write	10	Outer-shareable	YES
	Back	11	System	NO

4.10 AHB-AP

The AHB-AP implements the MEM-AP architecture to directly connect to an AHB-based memory system. Connection to other memory systems is possible through suitable bridging.

As part of the MEM-AP description, the AHB-AP has a number of implementation-specific features that are described in:

- *External interfaces*.
- *Implementation features* on page 4-31.
- *DAP transfers* on page 4-32.
- *Differentiation between system and access port initiated error responses* on page 4-33.
- *Effects of resets* on page 4-33.

For information about all the registers and features in a MEM-AP, see the *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*.

4.10.1 External interfaces

The primary external interface to the system is an AHB-Lite master port that supports:

- AHB in AMBA v2.0.
- ARM11 AMBA extensions.
- TrustZone extensions.

The AHB-Lite master port does not support:

- BURST and SEQ.
- Exclusive accesses.
- Unaligned transfers.

Table 4-9 shows the other AHB-AP ports.

Table 4-9 Other AHB-AP ports

Name	Type	Description
dbgen	Input	Enables AHB-AP transfers if HIGH. Access to the AHB-AP registers is still permitted if **dbgen** is LOW, but no AHB transfers are initiated. If a transfer is attempted when **dbgen** is LOW, then the DAP bus returns **dapslverr** HIGH.
spiden	Input	Permits secure transfers to take place on the AHB-AP. If **spiden** is HIGH, then **hprot[6]** can be asserted as programmed into the SProt bit in the CSW Register. See Chapter 3 *Programmers Model*.

HPROT encodings

hprot[6:0] is provided as an external port and is programmed from the Prot field in the CSW register with the following conditions:

- **hprot[4:0]** programming is supported.
- **hprot[5]** is not programmable and is always set LOW. Exclusive access is not supported, and therefore **hprot[2]** is not supported.
- **hprot[6]** programming is supported. **hprot[6]** HIGH denotes a non-secure transfer. **hprot[6]** LOW denotes a secure transfer. **hprot[6]** can be asserted LOW by writing to the SProt field in the CSW Register. A secure transfer can only be initiated if **spiden** is HIGH. If SProt is set LOW in the CSW Register to perform a secure transfer, but **spiden** is LOW, then no AHB transfer takes place.

See Chapter 3 *Programmers Model* for values of the Prot field.

HRESP

hresp[0] is the only RESPONSE signal that the AHB-AP requires:

- AHB-Lite devices do not support SPLIT and RETRY and therefore **hresp[1]** is not required. It is still provided as an input, and if not present on any slave it must be tied LOW. Any **hresp[1:0]** response that is not `0b00`, OKAY, is treated as an ERROR response.
- **hresp[2]** is not required because exclusive accesses are not supported in the AHB-AP.

HBSTRB support

hbstrb[3:0] signals are automatically generated based on the transfer size **hsize[2:0]** and **haddr[1:0]**. Byte, half-word and word transfers are supported. It is not possible for you to directly control **hbstrb[3:0]**.

Unaligned transfers are not supported. Table 4-10 shows an example of the generated **hbstrb[3:0]** signals for different-sized transfers.

Table 4-10 Example generation of byte lane strobes

Transfer description	haddr[1:0]	hsize[2:0]	hbstrb[3:0]
8-bit access to `0x1000`	`0b00`	`0b000`	`0b0001`
8-bit access to `0x1003`	`0b11`	`0b000`	`0b1000`
16-bit access to `0x1002`	`0b10`	`0b001`	`0b1100`
32-bit access to `0x1004`	`0b00`	`0b010`	`0b1111`

AHB-AP transfer types and bursts

The AHB-AP cannot initiate a new AHB transfer every clock cycle because of the additional cycles required to serial scan in the new address or data value through a debug port. The AHB-AP supports two **htrans** transfer types, IDLE and NONSEQ:

- When a transfer is in progress, it is of type NONSEQ.
- When no transfer is in progress and the AHB-AP is still granted the bus, the transfer is of type IDLE.

The only unpacked **hburst** encoding supported is SINGLE. Packed 8-bit transfers or 16-bit transfers are treated as individual NONSEQ, SINGLE transfers at the AHB-Lite interface. This ensures that there are no issues with boundary wrapping, to avoid additional AHB-AP complexity.

A full AHB master interface can be created by adding an AHB-Lite to AHB wrapper to the output of the AHB-AP, as provided in the *AMBA Design Kit*.

4.10.2 Implementation features

The AHB-AP provides the following specific MEM-AP features:

- Auto-incrementing of the Transfer Address Register with address wrapping on 1KB boundaries.
- Word, half-word and byte accesses to devices present on the AHB memory system.

- Packed transfers on sub-word transfers.

The AHB-AP does not support the following MEM-AP features:

- Big-endian. All accesses performed as expected to be to a little-endian memory structure.
- Slave memory port disabling. The AHB-Lite master interface is not shared with any other connection so there is no slave port to disable access to this interface. If the memory-map presented to the AHB-AP is to be shared with another AHB-Lite master then this is implemented externally to the DAP.

4.10.3 DAP transfers

This section describes:

- *DAP transfer aborts*.
- *Error response generation*.

DAP transfer aborts

The AHB-AP does not cancel the system-facing operation and returns **dapready** HIGH one cycle after **dapabort** has been asserted by the driving debug port. The externally driving AHB master port does not violate the AHB protocol. After a transfer has been aborted, the CSW Register can be read to determine the state of the transfer in progress bit, **TrInProg**. When **TrInProg** returns to zero, either because the external transfer completes, or because of a reset, the AHB-AP returns to normal operation. All other writes to the AHB-AP are ignored until this bit is returned LOW after a transfer abort.

Error response generation

This section describes:

- *System initiated error response*.
- *Access port initiated error response*.
- *AHB-AP reads after an abort*.
- *AHB-AP writes after an abort* on page 4-33.

System initiated error response

An error response received on the system driving master propagates onto the DAP bus when the transfer is completed. This response is received by the debug ports.

Access port initiated error response

Access port initiated error responses are:

- *AHB-AP reads after an abort*.
- *AHB-AP writes after an abort* on page 4-33.

AHB-AP reads after an abort

After a **dapabort** operation is carried out, and an external transfer is still pending, that is, the **TrInProg** bit remains HIGH, reads of all registers return a normal response except for reads of the Data R/W Register and banked registers. Reads of the Data R/W Register and banked registers return an error response because they cannot initiate a new system read transfer until the TrInProg bit in the CSW Register is cleared either by completing the system transfer or by a reset.

AHB-AP writes after an abort

After a **dapabort** operation is carried out, and an external transfer is still pending, that is, the transfer in progress bit remains HIGH, all writes to the access port return an error response, because they are ignored until the **TrInProg** bit is cleared.

4.10.4 Differentiation between system and access port initiated error responses

If **dapslverr** is HIGH and **TrInProg** is LOW in the CSW Register, the error is from either:

- A system error response if **dbgen** and **spiden** permit the transfer to be initiated.
- An AHB-AP error response if **dbgen** and **spiden** do not permit the transfer to be initiated.

Table 4-11 shows the error responses.

Table 4-11 Error responses with DAPSLVERR HIGH and TrInProg LOW

SProt	SPIDEN	DBGEN	Error response from	Reason
x	x	0	AHB-AP	No transfers permitted
0	0	1	AHB-AP	Secure transfers not permitted
0	1	1	System	Secure transfer produced an error response
1	x	1	System	Non secure transfer produced an error response

If **dapslverr** is HIGH and **TrInProg** is HIGH, then the error is from an access port error response. The transfer has not been accepted by the access port. This case can only occur after an abort has been initiated and while the system transfer has not completed.

4.10.5 Effects of resets

Assert the **hresetn** signal LOW at any time. The DAP interface must return **dapslverr** for the current transaction.

The **dapresetn** signal must only be asserted LOW when there is no pending transaction on the AHB interface.

4.11 APB-AP

The APB-AP implements the MEM-AP architecture to connect directly to an APB based system. The intention is that this bus is dedicated to CoreSight and other debug components.

As part of the MEM-AP description, the APB-AP has a number of implementation-specific features. These are described in:

- *External interfaces*.
- *Implementation features*.
- *DAP transfers*.

For information on all the registers and features in a MEM-AP, see the *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*

4.11.1 External interfaces

The primary interface on APB-AP is an APB AMBAv3 compliant interface supporting:

- Extended slave transfers.
- Transfer response errors.

Table 4-12 shows the other APB-AP ports.

Table 4-12 APB-AP other ports

Name	Type	Description
pdbgswen	Output	Enables self-hosted access to the debug APB at the APB multiplexer.
deviceen	Input	Disables device when LOW.

4.11.2 Implementation features

The APB-AP provides the following specific MEM-AP features:

- Auto-incrementing of the Transfer Address Register with address wrapping on 1KB boundaries.
- Slave memory port disabling a slave interface is provided through the APB interconnect to enable another APB master to connect to the same memory-map as the APB-AP.

The APB-AP does not support the following MEM-AP features:

- Big-endian. All accesses must be to a little-endian memory structure.
- Sub-word transfers. Only word transfers are supported.

The APB-AP supports a synchronous APB interface. The internal DAP interface and the APB interface operate from **dapclk**.

The APB-AP has one clock domain, **dapclk**. It drives the complete APB-AP. This must be connected to **pclkdbg** for the APB interface.

dapresetn resets the internal DAP interface and the APB interface.

4.11.3 DAP transfers

This section describes DAP transfers.

Effects of DAPABORT

The APB-AP does not cancel the system-facing operation, and returns **dapready** HIGH one cycle after **dapabort** is asserted by the debug port. The externally driving APB master port does not violate the APB protocol. After a transfer is aborted, the Control and Status Register can be read to determine the state of the transfer in progress bit, **TrInProg**. When **TrInProg** returns to zero, after completing the external transfer or on a reset, the APB-AP returns to normal operation. All other writes to the APB-AP are ignored until the TrInProg bit is returned LOW after a transfer Abort.

APB-AP error response generation

APB-AP error response generation is described in:

- *System initiated error response*.
- *AP-initiated error response*.
- *Differentiation between System-initiated and AP-initiated error responses*.

System initiated error response

An error response received on the APB master interface propagates onto the DAP bus when the transfer is completed. This is received by the debug ports.

AP-initiated error response

- APB-AP reads after an abort:

  After a transfer abort operation is carried out, and an external transfer is still pending, that is, the **TrInProg** bit in the CSW Register remains HIGH. Reads of all registers return a normal response except for reads of the Data R/W Register and banked registers. Reads of the Data R/W Register and banked registers return an error response because they cannot initiate a new system read transfer until the **TrInProg** bit in the CSW Register is cleared either by completing the system transfer or by a reset.

- APB-AP writes after an abort:

  After a Transfer Abort operation is carried out, and an external transfer is still pending, that is, the transfer in progress bit remains HIGH. All writes to the access port return an error response, because they are ignored until the **TrInProg** bit has cleared.

Differentiation between System-initiated and AP-initiated error responses

If **dapslverr** is HIGH and **TrInProg** is LOW, then the error is from a system error response.

If **dapslverr** is HIGH and **TrInProg** is HIGH, then the error is from an access port error response. The transfer has not been accepted by the access port. This case can occur after an abort has been initiated and while the system transfer has not completed.

4.12 APB Interconnect

The APB interconnect includes the following features:

The master or slave naming of the APB interfaces in the component is done from the APB interconnect component. The interface that responds to accesses is named the slave interface. In the APB interconnect, this corresponds to the interfaces that connect to external APB masters. The interface that initiates accesses is named the master interface. In the APB interconnect, this corresponds to the interfaces that connect to the external debug APB slaves.

4.12.1 Clock and reset

This component has a single clock domain, **clk**, driven by the debug APB clock. The master and slave interfaces operate on the same clock, **clk**.

This component has a single reset input, **resetn**, that is an asynchronous active-LOW reset input.

4.12.2 Functional interfaces

The APB interconnect with the ROM table has a configurable number of AMBA3 APB-compliant slave interfaces. Similarly, it has a configurable number of AMBA3 APB-compliant master interfaces.

The DbgSwEnable bit in the APB-AP can be used to prevent self-hosted, on-chip, accesses.

4.12.3 Device operation

This section describes device operation in the following subsections:

- *Accesses to ROM table*.
- *Arbitration*.
- *Error response* on page 4-37.
- *Address width on master interfaces* on page 4-37.
- *Address width on slave interfaces* on page 4-38.

Accesses to ROM table

Accesses to addresses in the range `0x0000` - `0x0FFC` are decoded to the ROM table. See the *ARM Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2* for information on ROM tables.

Arbitration

The internal arbiter arbitrates between competing slave interfaces for access to debug APB as the following algorithm demonstrates:

- When a slave interface raises a request, the highest priority is given to the slave interface with the lowest instance suffix, that is, `SlvIntf0 > SlvIntf1 > SlvIntf2 > … > SlvIntf(n-1)`. The order in which these slave interfaces raised their requests relative to each other is not used in arbitration.
- The arbitration is re-evaluated after every access.

Error response

The APB interconnect returns an error on its slave interface under any of the following conditions:

- The targeted debug APB device returns an error response.
- The address accessed by a slave interface does not decode to any debug APB device.
- A system access is attempted to a debug APB device when not permitted. This occurs when the DbgSwEnable bit in the APB-AP is cleared, and self-hosted, on-chip, accesses are attempted.

Address width on master interfaces

The width of the address bus on the master interface depends on the size of the address space allocated to that interface through bit[31] of the address bus. Bit[31] is always exported onto the master interface. Table 4-13 shows the address bus widths for each setting of size.

Table 4-13 Address width on master interfaces

Size of address space	Address bus on the master interface, where x=0 to NUM_MASTER_INTF-1
4KB	**paddrmx[11:2]**
8KB	**paddrmx[12:2]**
16 KB	**paddrmx[13:2]**
32 KB	**paddrmx[14:2]**
64 KB	**paddrmx[15:2]**
128 KB	**paddrmx[16:2]**
256 KB	**paddrmx[17:2]**
512 KB	**paddrmx[18:2]**
1 MB	**paddrmx[19:2]**
2 MB	**paddrmx[20:2]**
4 MB	**paddrmx[21:2]**
8 MB	**paddrmx[22:2]**
16 MB	**paddrmx[23:2]**
32 MB	**paddrmx[24:2]**
64 MB	**paddrmx[25:2]**
128 MB	**paddrmx[26:2]**
256 MB	**paddrmx[27:2]**
512 MB	**paddrmx[28:2]**
1 GB	**paddrmx[29:2]**

Address width on slave interfaces

Address width on the APB slave interface depends on the memory footprint occupied by the concatenation of all the defined master interfaces and the 4KB footprint of the ROM Table.

The number of address bits in the slave interface = `log2(Base address of Master_Interface_max + Size in bytes of that Master interface).[max = NUM_MASTER_INTF-1]`.

- Single-layer, bus-type interconnect. Supports up to four slave interfaces and up to 64 master interfaces.
- Tightly-coupled ROM Table at fixed base address of `0x00000000`.
- Support for cascading of APB interconnects and ROM tables.
- Single clock and power domains.
- Compliant with the AMBA3 APB protocol and the CoreSight architecture.
- 32-bit data bus for each APB interface.
- APB slave interfaces address bus width automatically scaled based on the APB master interfaces and memory map configuration.
- Address width on APB master interface depends on size of the address space allocated to each interface.
- Fixed priority arbitration.
- Control to enable or disable SoC system masters accessing the debug sub-system.

4.13 APB asynchronous bridge

The APB asynchronous bridge enables data transfer between two asynchronous clock domains. The APB asynchronous bridge is designed to exist across two power domains and provides an optional LPI.

4.13.1 Clock and reset

The clocks and resets of the APB asynchronous bridge are:

PCLKM Clock for master interface.

PCLKENM Clock enable for master interface.

PRESETMn Interface reset for master interface.

PCLKS Clock for slave interface.

PCLKENS Clock enable for slave interface.

PRESETSn Interface reset for slave interface.

4.13.2 Functional interface

The APB asynchronous bridge has the following interfaces:

- One APB master compliant with APB3.
- One APB slave compliant with APB3.
- One optional LPI slave.

4.13.3 Functional description

The APB asynchronous bridge permits the APB data to be transferred to another clock domain. This component also supports the power gating of the master side through the LPI present in the slave side if the LPI is configured during implementation.

The main modes of operations for the APB asynchronous bridge are:

Generation of PREADYS signal on the slave interface

For a transfer that is initiated by a master, the bridge extends the transfer by driving the **PREADYS** signal LOW on its slave interface, until the bridge master interface has completed the transfer to the target device. **PREADYS** is driven LOW by default. The bridge waits for **PREADYM** to be driven HIGH before indicating a completion of transfer on the slave interface by asserting **PREADYS** HIGH.

Low-power features

The APB asynchronous bridge supports an optional low-power interface to the power controller. The system-level power controller can request the master interface of the bridge to go into low-power state through the low power interface. The bridge will enter the low-power state when there is no pending transaction. When the bridge is in low-power mode, a wake-up request is generated on a new transaction, and the transaction completes when the power controller brings the master interface out of low-power state. The bridge stalls the transaction on the slave interface until the master interface is brought out of low-power state and ensures that there is no loss of data transferred through the bridge.

4.14 APB synchronous bridge

The APB synchronous bridge enables data transfer between two synchronous clock domains.

4.14.1 Clock and reset

The APB synchronous bridge operates in a single clock domain with one asynchronous reset. The clocks and resets of the APB synchronous bridge are:

- **PCLK**, the APB clock.
- **PRESETn**, the APB interface.

4.14.2 Functional interface

The APB synchronous bridge has the following interfaces:

- One APB master compliant with APB3.
- One APB slave compliant with APB3.
- One optional LPI port.

4.14.3 Functional description

APB synchronous bridge can be used as a register slice on the APB path

Normal operation

The assertion of **PSELS** indicates that a valid transaction is presented on the slave interface. The bridge forwards the control data to the master interface by keeping **PREADYS** LOW. **PREADYS** is asserted HIGH only after the bridge samples **PREADYM** HIGH on the master interface.

4.15 Auxiliary Access Port

An auxiliary interface is an interface on the DAPBUS Interconnect that does not connect to any of the supplied APs. An auxiliary interface can be connected to the access port of processors that have a compliant debug interface, such as the Cortex-M3 processor.

4.16 Authentication requirements for Debug Access Port

This section describes the functionality that is available in the debug and trace components to permit authentication using the signals, and describes how they are connected. If you do not require the system to support this level of control, you can simplify the system design.

APB-AP has one authentication signal, called **deviceen**:

- If the APB-AP is connected to a debug bus, **deviceen** must be tied HIGH.
- If the APB-AP is connected to a system bus dedicated to the secure state, this signal must be connected to **spiden**.
- If the APB-AP is connected to a system bus dedicated to the non-secure state, this signal must be connected to **dbgen**.

For more information, see the *CoreSight Architecture Specification*.

4.17 Clocks, power, and resets

Implement the DAP to use multiple clock domains. Synchronous bridges must be instantiated when the clock domains of interacting components are edge-aligned but of different frequencies. Asynchronous bridges have to be instantiated when the clock domains of interacting components are asynchronous.

The SWJ-DP must be present in the always on power domain.

Bridges with LPI must be instantiated where there is a power domain boundary between interacting components.

Chapter 5
ATB Interconnect Components

This chapter describes the ATB interconnect components. It contains the following sections:

5.1 ATB replicator

The ATB replicator propagates the data from a single ATB master to two ATB slaves at the same time.

Figure 5-1 shows an example ATB replicator.

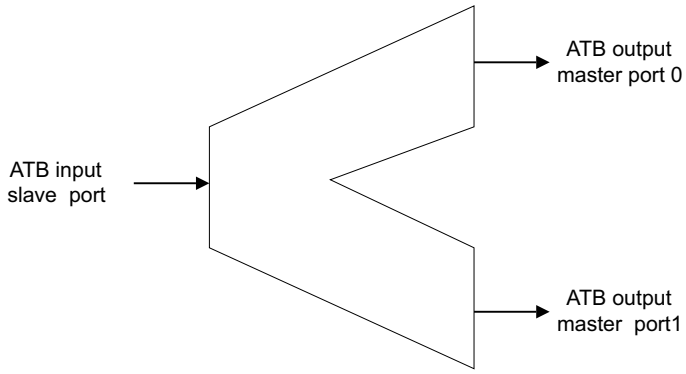


Figure 5-1 Example ATB replicator

5.1.1 Clock and reset

The ATB replicator contains only one clock domain. The block uses **RESETn** to reset all the flip flops in the design. The clock and reset signals of the ATB replicator are:

PCLKENDBG Clock enable for an optional debug APB port.

CLK ATB clock.

RESETn ATB reset.

5.1.2 Functional interfaces

The ATB replicator has a single slave ATB port, two ATB master ports, and one optional APB port.

5.1.3 Functional overview

The ATB replicator permits the connection of two trace sinks. Buses cannot be connected directly together.

Any connected blocks, on both the inputs and the outputs, must operate on the same **CLK** domain.

Trace data flow

As data is received from the trace source, it is passed on to all trace sinks at the same time. The replicator does not accept more data from the trace source until all the trace sinks have accepted this data. This has the impact of reducing the throughput of the replicator to match that of the slowest trace sink.

If the replicator is programmed to be configurable, this adds the ability to ignore specific trace sinks so that they are unable to influence the throughput to other ports.

Flushing AFVALIDM and AFREADYM

Whenever one of the trace sinks initiates a flush to remove old information from the system, the replicator must propagate this request even when the other sink has not requested it.

This does not cause any problems with the non-requested trace sink. It receives the flushed data, if a request to start a flush is received from the other trace sink. When there is already a flush operation under way that was begun by the first trace sink, then the replicator must wait until the first request is serviced, and then service the second request.

SYNCREQ

The synchronization merging logic generates synchronization requests at a rate corresponding to the most frequent of the input requests from all active trace sinks.

ID Filtering

When the APB programmable interface is configured, the bit in the IDFILTERx Register determines whether the ATB transactions with the corresponding ID is discarded or passed to the ATB master port <x>. On reset, these IDFILTERx Registers are reset to 0, so that all ATB transactions are passed to both ATB master ports. There is no limitation on the IDFILTERx Register programming. The programmed value takes effect when the ongoing ATB transactions are completed, or when the APB programming is completed, if there are no ongoing transactions.

5.2 ATB funnel

The ATB funnel component merges multiple ATB buses into a single ATB bus.

5.2.1 Clock and reset

The optional APB interface has a clock enable **PCLKENDBG**. The ATB funnel has a single clock domain with clock input, **CLK**.

The funnel has a single reset input, **RESETn**.

The **RESETn** signal is an asynchronous active-LOW reset.

5.2.2 Functional interface

The ATB funnel has a configurable number of ATB slave interfaces and one ATB master interface.

The slave ATB interfaces connect to replicators, trace sources, trace links, or any other component with a standard ATB master.

The master interface connects to replicators, trace sinks, trace links, or any other component with a standard ATB slave.

All ATB interfaces are configurable for the ATB DATA width.

5.2.3 Funnel functionality

This section describes the functionality of the ATB funnel.

Port enable

Set or clear the appropriate enable slave port bit in the Control Register to enable or disable the corresponding ATB slave interface.

Disabling of the ATB slave is effective at the transaction boundary when **ATREADYSx** is set to 1. If there is no valid transaction on the slave port, the disable is effective immediately.

You can enable or disable the ATB slaves at any time without any restriction with regard to the state of the trace system.

In configurations with no programmers model, all ATB slaves are enabled and cannot be disabled.

5.2.4 Arbitration

The funnel combines trace streams from multiple ATB buses into a single ATB bus. Streams are combined by switching between ATB masters connected to funnel inputs.

The selection of the ATB slave is performed using a priority-based arbitration scheme.

However, the following adverse consequences associated with frequent switching exist:

- Loss of efficiency in the formatter, resulting in loss of trace bandwidth downstream of a formatter.
- Loss of bus efficiency in the upsizer.

To alleviate the problems associated with frequent switching, the funnel implements a minimum hold time feature.

Minimum hold time

When the bus is granted to a particular trace source, an ATB master, it stays granted for a programmed number of sequential transfers. The number of sequential transfers that a master can perform without losing the ATB bus to a higher priority source is the minimum hold time. If there are no more pending transfers, the bus is granted to another master even if the hold time number has not been exceeded. If the other masters do not have valid data after the hold time has expired, the same master is granted the bus again, but the hold counter is not re-loaded with the value from the register. This enables other masters to obtain a grant as soon as they have data.

The minimum hold time is set by the value of the HT field in the Control Register plus 1. The reset value of the HT field is `0x3` meaning a minimum hold time of 4.

The maximum value for the minimum hold time is `0xE`, and equates to 15 transactions.

The hold time value in the Control Register can be programmed at any time, without any restriction with regard to the state of the trace system.

The new hold time value takes effect from the next arbitration point after the hold time has been changed.

The hold time value of `0xF` is reserved and programming the hold time with `0xF` is considered to be a programming error.

Example minimum hold time waveform

Figure 5-2 shows the effect of minimum hold time for a two ATB slave port funnel programmed as follows:

- Slave port priority arbitration, where slave port 0 has the higher priority, and slave port 1 has the lower priority.
- A minimum hold time of 4.

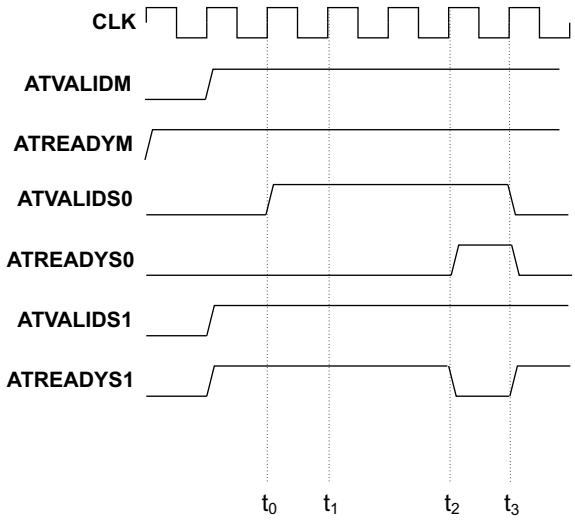


Figure 5-2 ATB funnel minimum hold time example

Table 5-1 shows the sequence of events in Figure 5-2 on page 5-5.

Table 5-1 Event sequence

Time	Event
t_0	• Slave port 1 is currently selected. • Higher priority slave port 0 has a pending transfer. • Slave port 1 remains selected because the minimum hold time has not expired.
t_1	• Slave port 1 remains selected because the minimum hold time has not expired.
t_2	• Minimum hold time expires for slave port 1 and funnel switches to slave port 0.
t_3	• Slave port 0 has no more data to transfer and the funnel switches back to slave port 1.

Priority setting

The funnel implements programmable priority for the attached ATB masters. Priority values for each ATB slave interface are defined in a 3-bit field in the Priority Control register. A port programmed with the value 0 gets the highest priority. A port programmed with the value 7 gets the lowest priority. At reset, all ports have a value of 0.

The bandwidth sharing scheme is used when selecting the next funnel input among masters with equal priority. The priority value for the ATB slave interface is only changed when the ATB slave interface is disabled. Programming the priority value for the enabled ATB slave is considered to be a programming error. The funnel takes the newly programmed priority values at the arbitration point.

Additional consideration for ATID changes

Frequent funnel switching results in a trace stream with frequent ATID changes leading to inefficiency in the formatter. In the case of cascaded funnels, even with hold time implemented, some unnecessary ATID changes can still be present in the trace stream.

To minimize ATID switching, transfers from a single funnel input are considered sequential only if they have the same ATID.

An ID change overrides hold time and causes the funnel to switch to another master with pending transfers.

Arbitration scheme

The funnel implements a round robin arbitration scheme.

A new arbitration is performed at every arbitration point. An arbitration point occurs when no sources were selected on the previous cycle or when the state of the previously selected source is:

- Its hold time has expired.
- Its **ATID** has changed from the previous cycle.
- Its **ATVALID** signal is LOW.
- Its input has been disabled.
- Its flush has been acknowledged after transfer has been completed.

At an arbitration point, first identify all the sources with valid data. If no source has valid data, the next cycle is an arbitration point and no source is selected. Otherwise, the following criteria is used to select the valid source, in decreasing order of priority:

1. Sources in the flush state.
2. Sources with a higher programmed priority level.
3. Sources which were not previously selected.
4. Sources with a lower source number.

Note

Hold time is reloaded at each arbitration point.

Example arbitration waveform

The following are the funnel configurations:

- Four ATB slave interfaces.
- Slave port priority. Slave port 0 and slave port 1 are the highest priority, slave port 3 is the lowest priority.
- Minimum hold time is 2.

Figure 5-3 shows an example waveform for arbitration.

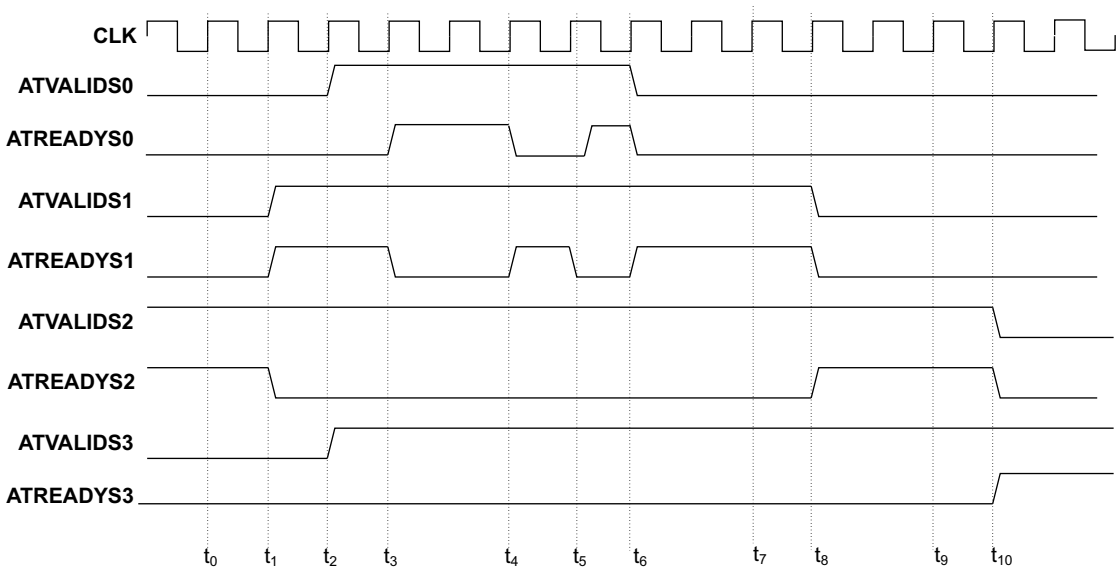


Figure 5-3 ATB funnel arbitration example

Note

ATIDS1 has changed since the previous cycle.

Table 5-2 shows the sequence of events in Figure 5-3 on page 5-7.

Table 5-2 Event sequence

Time	Event
t_0	• Slave port 2 is currently selected.
t_1	• Funnel switches to high priority slave port 1.
t_2	• High priority slave port 0 asserts **ATVALIDS0**. • Lowest priority slave port 3 asserts **ATVALIDS3**.
t_3	• Minimum hold time expires for slave port 1 and the next highest priority port, slave port 0, is selected.
t_4	• Minimum hold time expires for slave port 0 and the next highest priority port, slave port 1, is selected.
t_5	• **ATID** changes for slave port 1, minimum hold time is overridden so the next highest priority port, slave port 0, is selected.
t_6	• Slave port 0 finishes. • Funnel switches to the next highest priority port, that is, back to slave port 1.
t_7	• Minimum hold time expires for slave port 1, but being highest priority it remains selected.
t_8	• Slave port 1 finishes. • Funnel switches to next highest priority port, that is, slave port 2.
t_9	• Minimum hold time expires for slave port 2, but being higher priority than slave port 3, it remains selected.
t_{10}	• Slave port 3 finishes. • Funnel switches to the last remaining port, slave port 3.

5.2.5 Flushing

The funnel implements the flush mechanism. When a flush request is received at the master port of the funnel, all slave ports request a flush together to ensure that all historical trace data is collected. The funnel acknowledges the flush only after all slave ports have completed their flush sequence.

The funnel can assert **AFREADYM** HIGH when all slave ports have returned **AFREADYS** HIGH.

To enable devices to drain at different rates, for example, if they were on different clock domains, when the funnel is flushing trace sources, the arbitration process can still be in place. This method stops high-priority sources from locking the system if they take a number of cycles to present valid data. Under normal operation, if a high-priority source does not present any valid data, then the next highest priority source that does have valid data is selected.

Example flushing waveform

The following are the funnel configurations:

- Four ATB slave interfaces.
- Slave port priority, slave port 0 is the highest priority, and slave port 3 is the lowest priority.
- Minimum hold time is 2.

Figure 5-4 on page 5-9 shows an example waveform for arbitration.

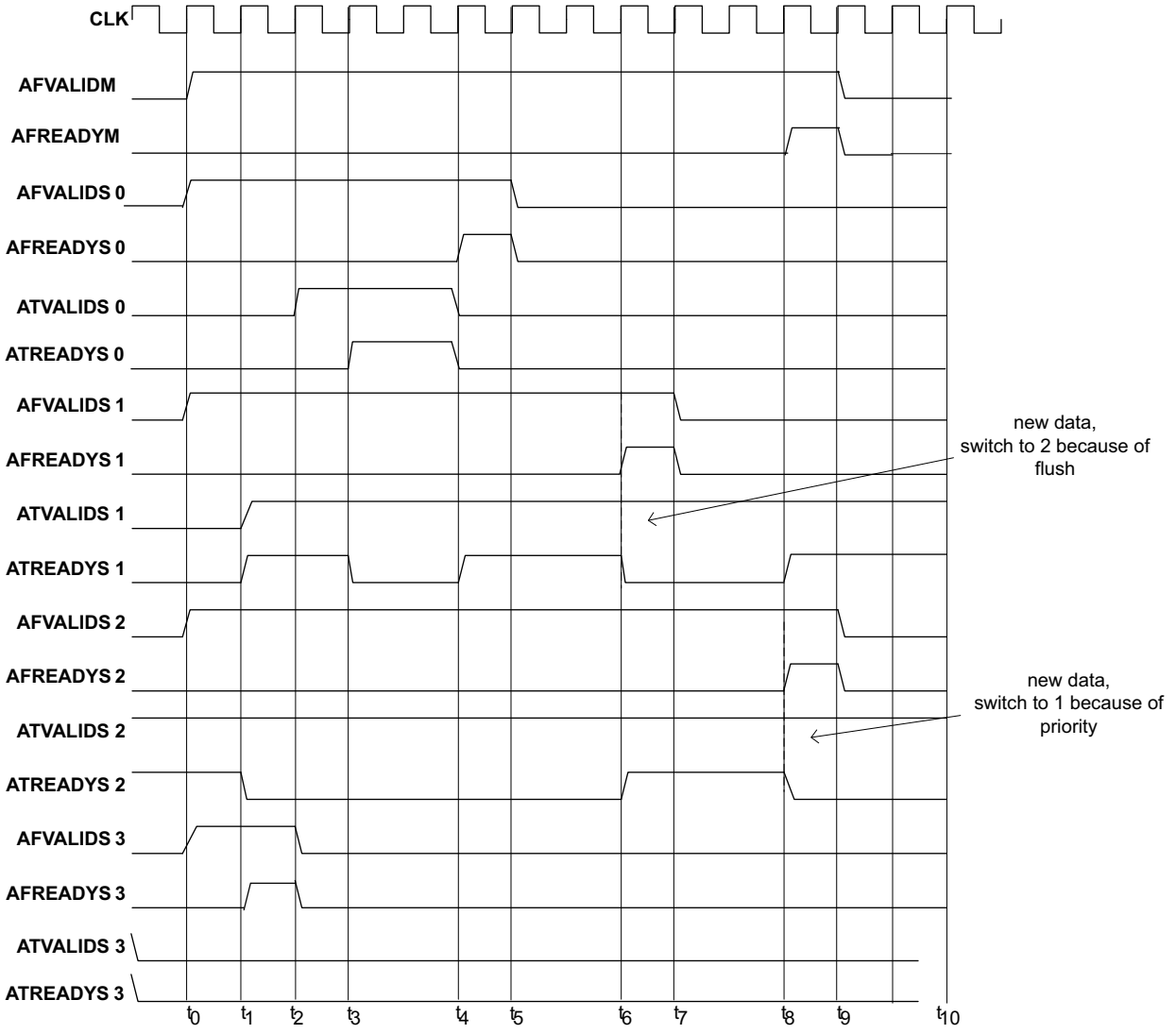


Figure 5-4 ATB funnel example flushing waveform

Table 5-3 shows the sequence of events in Figure 5-4.

Table 5-3 Event sequence

Time	Event
t_0	• A request from the trace sink device to flush the system exists.
t_1	• The flush request is propagated onto slave ports. • The Funnel selects slave port 1 to drain first.
t_2	• Slave port 3 returns **AFREADYS3** and this causes **AFVALIDS3** to be de-asserted in the next cycle.
t_3	• The minimum hold time expires for slave port 1 and the funnel switches to slave port 0.
t_4	• Slave port 0 has finished with flushed data. • Draining of slave port 1 continues.
t_5	• **AFREADYS0** goes HIGH.

Table 5-3 Event sequence (continued)

Time	Event
t_6	• Slave port 1 has finished with flushed data. • Slave port 1 continues with new data. • The funnel selects the last remaining slave in the flush state, slave port 2.
t_7	• **AFREADYS1** goes HIGH.
t_8	• Slave port 2 has finished with flushed data. • Slave port 2 continues with new data. • The funnel selects slave port 1 because it is a higher priority port.
t_9	• **AFREADYS2** goes HIGH • All slave ports have responded with **AFREADY** HIGH to indicate that there is no more historical information remaining at this level. This is indicated on **AFREADYM** with a HIGH response, and this in turn results in **AFVALIDM** returning LOW. • The flush sequence is now complete.

5.2.6 Syncreq propagation

The funnel can propagate ATB synchronization requests, SYNCREQ, received on the ATB master port to all enabled ATB slave ports.

5.2.7 Configuration

The funnel is configured using the debug APB interface.

5.2.8 Cascaded funnel support

The funnel is a combinatorial block, and when a cascaded funnel configuration is implemented, a register slice that is a forward, reverse, or full register slice, must be instantiated between the cascaded funnels to avoid combinatorial timing loops.

5.2.9 Unlocking funnel access

The Lock Status Register and Lock Access Register control access to the CoreSight funnel to ensure that the possibility of accidental access is extremely small.

The funnel implements a 32-bit LAR.

5.2.10 Claim scheme

To facilitate cooperation of debug, the funnel implements a CoreSight claim scheme. The Claim Tag registers provide 4 bits that can be separately set and cleared to indicate the current owner of the funnel.

5.2.11 Topology detection

The funnel supports topology detection through a set of integration registers that enable reading or writing **ATVALID** and **ATREADY** of all the ATB interfaces. Integration mode is enabled by setting the Integration mode bit in the ITCTRL Register.

Register ITATBCTR2 is the Integration Test ATB Control 2 Register. A write to ITATBCTR2 outputs data on **ATREADYSn**, where n is defined by the status of the Funnel Control Register. A read from ITATBCTR2 returns the data from **ATREADYM**.

Register ITATBCTR0 is the Integration Test ATB Control 0 Register. A write to ITATBCTR0 sets the value of **ATVALIDM**. A read from ITATBCTR0 returns the value of **ATVALIDSn**, where n is defined by the status of the Funnel Control Register.

It is illegal to have more than one ATB slave port enabled while in integration mode and performing topology detection. No hardware protection exists and enabling multiple ports is considered to be a programming error.

After performing integration or topology detection, you must reset the system to ensure the correct behavior of CoreSight and other connected system components that are affected by the integration or topology detection.

5.2.12 Fixed configuration funnel

In a fixed configuration, the funnel cannot have any user-programmable registers and it cannot have an APB slave interface.

The fixed configuration funnel supports the following features:

- All ATB slave ports are enabled.
- All ATB slave ports have a priority defined by the port number, and a round-robin scheme is implemented if the corresponding configuration option is selected.
- The hold time is set to four transfers, that is, the HT field value is set to `0x3`.
- The funnel is transparent for topology detection.

5.3 ATB upsizer

The ATB upsizer combines narrow trace data into wider data.

5.3.1 Clocks and reset

The bridge operates on a single clock **CLK**.

RESETn is used as an asynchronous reset across the entire design.

5.3.2 Functional interface

The ATB upsizer has a slave port of narrow width and a master port of wider width. The widths of the master and slave ports are configurable.

5.3.3 Component functionality

The following describes the functionality of the ATB upsizer:

Normal operation

The ATB upsizer merges transfers from the narrow ATB slave interface, and outputs buffered data on a wider master interface.

The ATB upsizer preserves the ATB features, LSB-aligned, and packed data.

The following are the rules for transfer merging:

- Merge the transfers with same **ATID**. The **ATID** change stops the merging process.
- Data is merged until:
  - The wide ATB bus data word is completed
  - Less than the full narrow ATB data bus is received.
- When data is part of a flush, flush acknowledges, **AFVALIDS**, and **AFREADYS** can end the merging process.
  - Any data transferred along with a flush acknowledge must not be merged with the flush data.

ATREADYS and ATVALIDM generation

The upsizer data path consists of wide and narrow buffers.

Data from a slave port is written to a wide buffer until:

- The wide buffer is full.
- There is a merge stop event such as an **ATID** change.

The **ATREADYS** signal is HIGH when an upsizer is ready to accept new data, that is, until data is written to a narrow buffer. The **ATVALIDM** is asserted HIGH after:

- The wide buffer is filled.
- There is a merge stop event such as an **ATID** change.

ATBYTES calculation

The ATB upsizer calculates the master side **ATBYTES** to match the number of bytes in the output transfer.

Example 1:4 waveform

The ATB upsizer parameters are:

- ATB slave **ATDATA** is 32-bit.

- ATB master **ATDATA** is 128-bit.

Figure 5-5 shows the example waveform sequence of the 1:4 upsizer. This example assumes that the following statements apply throughout the example:

- **ATIDS** remains constant.
- No flush requests are received.

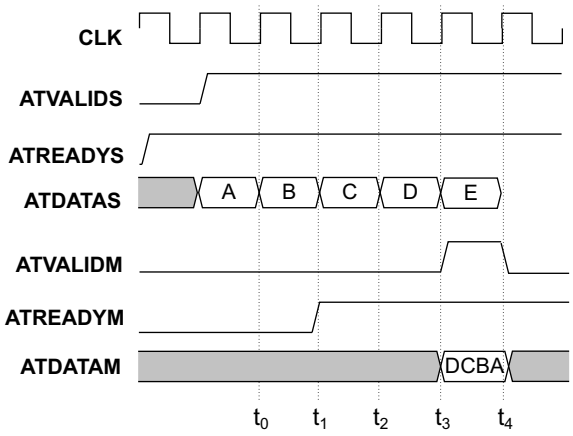


Figure 5-5 Upsizer example waveform

Table 5-4 shows the sequence of events in Figure 5-5.

Table 5-4 Upsizer example description

Time	Event	
t_0	•	Least significant data beat arrives at the upsizer input.
t_1	•	Second least significant data beat arrives at the upsizer input.
t_2	•	Third least significant data beat arrives at the upsizer input.
t_3	•	Most significant data beat arrives at the upsizer input.
t_4	•	All data beats are concatenated and output from the upsizer.

5.3.4 Flush

The ATB upsizer implements the flush mechanism.

On assertion of **AFVALIDM** to the ATB upsizer master port, the **AFVALIDS** signal is asserted on ATB slave port. **AFVALIDS** must then remain asserted until the slave port responds with **AFREADYS**.

The ATB upsizer can assert **AFREADYM** HIGH, when the slave port has returned **AFREADYS**, and transmitted all flush data within the upsizer.

Example flushing waveform

Figure 5-6 on page 5-14 shows the example flushing waveform sequence.

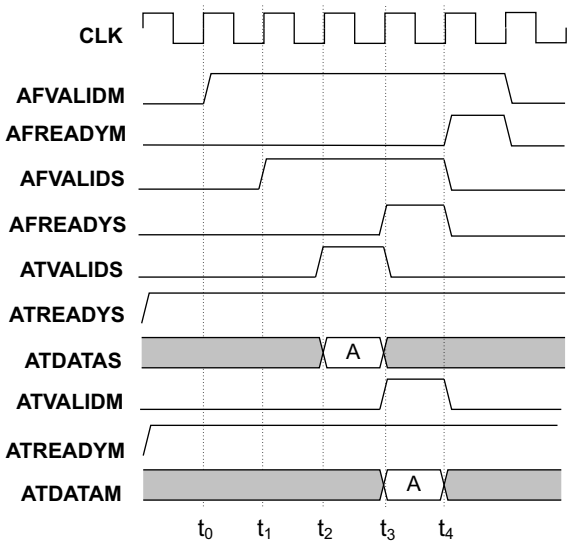


Figure 5-6 Example flushing waveform

Table 5-5 shows the example flushing waveform sequence.

Table 5-5 Example flushing waveform

Time	Event
t_0	The master port, **AFVALIDM**, is asserted.
t_1	The slave port, **AFVALIDS**, is asserted in the next clock cycle.
t_2	The last data on the slave port is accepted.
t_3	• The slave port, **AFREADYS**, is asserted. • The **ATVALID** is asserted on the master port. • The last data is accepted on the master port.
t_4	The master port, **AFREADYM**, is asserted.

5.3.5 Syncreq propagation

The ATB upsizer propagates the ATB synchronization requests, **SYNCREQ**, received from the ATB master port to the ATB slave port.

5.4 ATB downsizer

The ATB downsizer splits the wide trace data into multiple beats of narrow data.

5.4.1 Clocks and reset

The ATB downsizer has a single clock domain with clock input, **CLK**. The ATB downsizer has a single reset input, **RESETn**.

RESETn is used as the asynchronous active LOW reset.

5.4.2 Functional interface

The ATB downsizer has one ATB slave interface and one ATB master interface.

The slave ATB interface connects to replicators, trace sources, and trace links, or any other component with a standard ATB master.

The master interface connects to replicators, trace sinks, and trace links, or any other component with a standard ATB slave.

5.4.3 Component functionality

The following describes the functionality of the ATB downsizer:

Normal operation

The downsizer splits transfers from the wide ATB slave interface, and outputs them as multiple transfers on a narrower master interface.

The ATB master data remains LSB-aligned and packed.

Splitting rules

The rules for transfer split are:

- Break the transfer into a number of smaller sub-transfers based on **ATBYTESS**.
- The ATIDM of each sub-transfer is the same as the original transfer **ATIDS**.
- The **ATBYTESM** is calculated for each sub-transfer.

ATREADYS and ATVALIDM generation

The ATB downsizer data path consists of a data multiplexer.

A transaction on the slave port is held until all sub-transfers are finished on the master port. The **ATREADYS** signal goes HIGH when the last **ATREADYM** signal goes HIGH.

The **ATVALIDM** signal is asserted when the **ATVALIDS** signal is asserted. It stays asserted until the last sub-transfer is acknowledged with **ATREADYM**.

ATBYTES calculation

The downsizer calculates the master side, **ATBYTESM**, to match the number of bytes in the output transfer.

Example 4:1 downsizer waveform

The ATB downsizer parameters are:

- ATB slave **ATDATA** is 32-bit.
- ATB master **ATDATA** is 8-bit.

Figure 5-7 shows an example waveform sequence of the ATB downsizer.

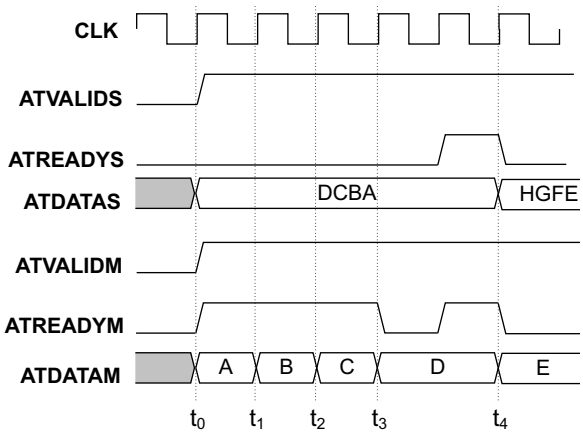


Figure 5-7 Downsizer example waveform

Table 5-6 shows the 4:1 downsizer example waveform sequence.

Table 5-6 Downsizer example waveform

Time	Event
t_0	• The transaction DCBA is initiated on the slave port. • The sub-transaction A is initiated on the master port.
t_1	• The sub-transaction A is accepted on the master port. • The sub-transaction B is initiated on the master port.
t_2	• The sub-transaction B is accepted on the master port. • The sub-transaction C is initiated on the master port.
t_3	• The sub-transaction C is accepted on the master port. • The sub-transaction D is initiated on the master port.
t_4	• The sub-transaction D is accepted on the master port. • The transaction DCBA is accepted on the slave port. • The transaction HGFE is initiated on the slave port. • The sub-transaction HGFE is initiated on the master port.

5.4.4 Flush

The ATB downsizer can implement the flush mechanism.

On assertion of **AFVALIDM** to the ATB downsizer master port, the **AFVALIDS** signal is asserted on the ATB slave port.

The ATB downsizer asserts **AFREADYM** when **AFREADYS** is asserted.

Example flushing waveform

Figure 5-8 on page 5-17 shows the example flushing waveform sequence.

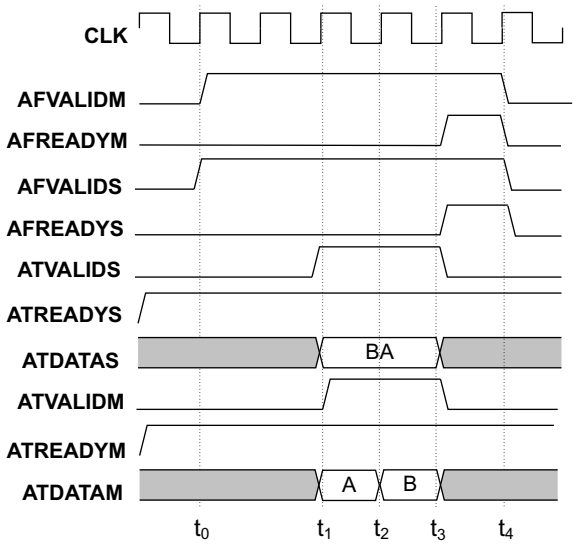


Figure 5-8 Downsizer example flushing waveform

Table 5-7 shows the example flushing waveform sequence.

Table 5-7 Example flushing waveform

Time	Event
t_0	• The **AFVALIDM** signal is asserted on the master port. • The **AFVALIDS** signal is asserted on the slave port.
t_1	• The flush data transfer BA is initiated on the slave port. • The sub-transfer A is initiated on the master port.
t_2	• The sub-transfer A is completed on the master port. • The sub-transfer B is initiated on the master port.
t_3	• The sub-transfer B is completed on the master port. • The transfer BA is completed on the slave port.
t_4	• The **AFREADYS** signal is asserted on the slave port. • The **AFREADYM** signal is asserted on the master port.

5.4.5 Syncreq propagation

The ATB downsizer propagates ATB synchronization requests, **SYNCREQ**, received from the ATB master port to the ATB slave port.

5.5 ATB asynchronous bridge

The ATB asynchronous bridge permits data transfer between two asynchronous clock domains. The ATB asynchronous bridge is designed to exist across two power domains, and provides an LPI.

This bridge has one input ATB interface and one output ATB interface. The input and the output interfaces are on different clock domains, and the bridge is referred to as an asynchronous bridge. The input side is referred to as the slave side of the bridge and the output side output is the master side of the bridge. The bridge has been implemented to enable independent powerdown of either side.

5.5.1 Functional interfaces

The ATB asynchronous bridge has the following functional interfaces:

- ATB slave interface.
- ATB master interface.
- An optional LPI interface.

5.5.2 Clock and reset

The ATB asynchronous bridge uses the following clock and reset signals:

CLKS Clock for slave interface.

RESETSn Reset for slave interface.

CLKENS Clock enable for the slave interface.

CLKM Clock for master interface.

RESETMn Reset for master interface.

CLKENM Clock enable for the master interface.

5.5.3 Device operation

The following are the operations of the ATB asynchronous bridge:

Normal operation

Figure 5-10 on page 5-19 shows an input example event sequence of normal operation for the ATB asynchronous bridge.

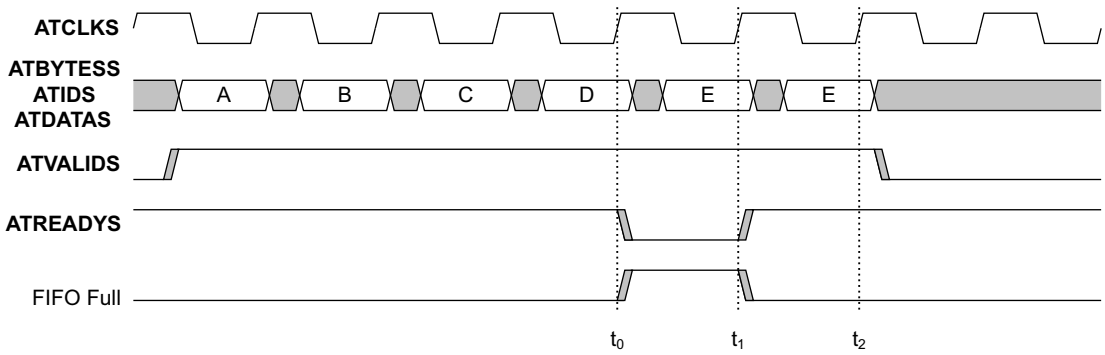


Figure 5-9 ATB asynchronous bridge normal operation for input

Table 5-8 shows an input event sequence of normal operation for the ATB asynchronous bridge.

Table 5-8 Event sequence for input example

Time	Event
t_0	• The FIFO is full. • The **ATREADYS** is driven LOW and the bridge can not accept additional data from the trace source.
t_1	• The FIFO is not full. • The **ATREADYS** is driven HIGH and the bridge can accept additional data from the trace source. • The data, E, is available for one additional clock cycle because of the wait state.
t_2	The data, E, accepted on the slave interface and stored in the FIFO.

Figure 5-10 shows an output example of normal operation for the ATB asynchronous bridge.

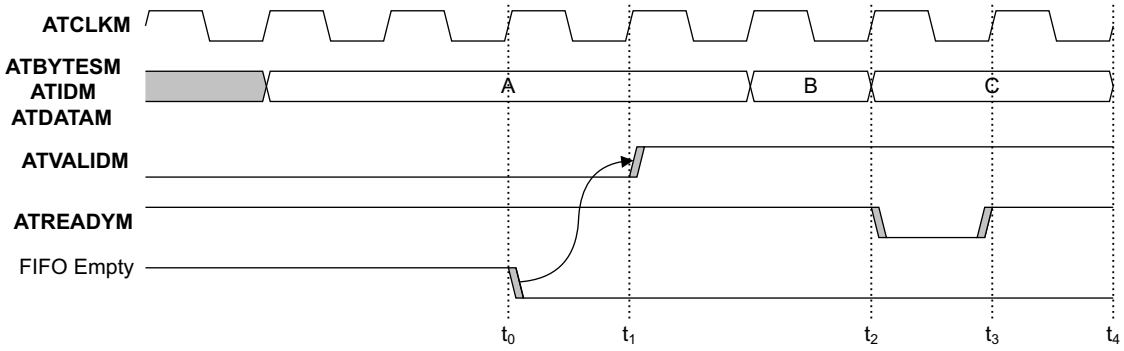


Figure 5-10 ATB asynchronous bridge normal operation for output

Table 5-9 shows an output event sequence of normal operation for the ATB asynchronous bridge.

Table 5-9 Event sequence for output example

Time	Event
t_0	The FIFO is no longer empty.
t_1	The data transfer is initiated on the master interface.
t_2	• The data, B, transfer is completed. • The data, C, transfer is initiated, but the downstream device indicates that it cannot complete the transfer by driving **ATREADYM** LOW.
t_3	• The data, C, transfer is extended because of wait state. • Downstream device indicates that it is ready to complete transfer.
t_4	The data, C, transfer is completed.

Flush operation

Figure 5-11 on page 5-20 shows the flush operation of the ATB asynchronous bridge.

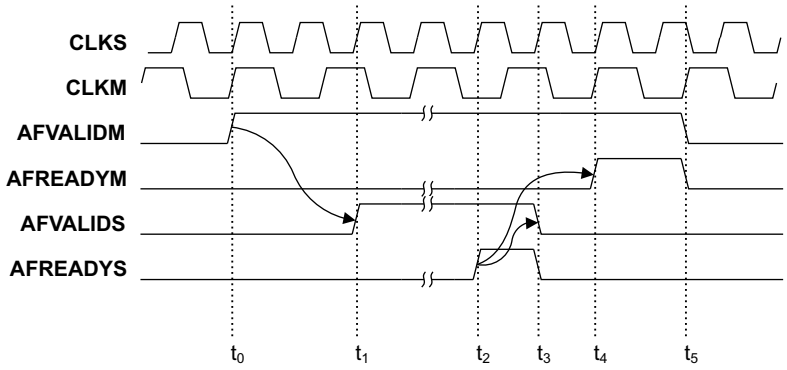


Figure 5-11 ATB asynchronous bridge flushing behavior

Table 5-10 shows the flush operation of the ATB asynchronous bridge.

Table 5-10 Event sequence of the ATB asynchronous bridge flush operation

Time	Event
t_0	The master interface receives a flush request.
t_1	The flush request propagated to the slave interface.
t_2	The slave interface receives an indication that the flush is complete.
t_3	The flush request de-asserted on the slave interface
t_4	The flush completion is indicated on the master interface by asserting **AFREADYM**
t5	The flush operation is completed on the master interface

The ATB asynchronous bridge is also flushed when a powerdown request occurs, that is, **CSYSREQ** is asserted LOW. The bridge cannot request a powerup so **CACTIVE** is always tied LOW.

Figure 5-12 shows the flush operation of the ATB asynchronous bridge because of a powerdown request.

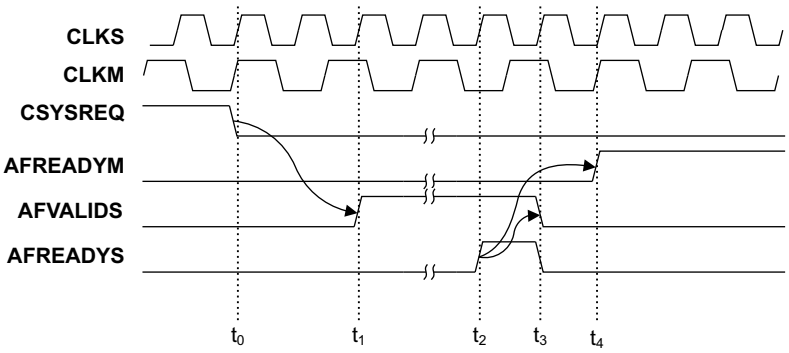


Figure 5-12 ATB asynchronous bridge flushing operation during powerdown request

Table 5-11 shows event sequence of the flush operation of the ATB asynchronous bridge because of a powerdown request.

Table 5-11 Event sequence of the ATB asynchronous bridge flushing operation during powerdown request

Time	Event
t_0	The powerdown request is detected.
t_1	The flush request is initiated on the slave interface by asserting **AFVALIDS**.
t_2	The flush completion is detected on the slave interface.
t_3	The flush operation is completed on the slave interface and the **AFVALIDS** is driven LOW.
t_4	The flush completion is indicated on the master interface by asserting **AFREADYM**.

5.5.4 Low-power features

The ATB asynchronous bridge is designed to support two power domains. The low-power interface provides the signal interface to the power controller. The ATB asynchronous bridge accepts low-power entry when it is requested from the system power controller.

The ATB asynchronous bridge does not initiate wake-up. Exit from the low-power state is always initiated by the system controller. The ATB asynchronous bridge initiates ATB flush before acknowledging low-power entry to drain relevant data from the system. When the flush is completed on the slave interface, the bridge acknowledges all transfers by asserting **ATREADYS** HIGH, but does not store the transfers in the bridge.

The low-power state is entered only when the buffers inside the bridge do not have valid data.

5.6 ATB synchronous bridge

The ATB synchronous bridge enables the data transfer between two synchronous clock domains. It also provides an LPI to support power-gating within a single voltage domain.

5.6.1 Clock and reset

The ATB synchronous bridge is in a single clock domain and contains only one clock input **CLK**. However, there are clock enables present in the master and slave interfaces to enable the ports to operate at different frequencies. The ATB synchronous bridge uses the following clock and reset signals:

CLK Clock.

RESETn Asynchronous active-LOW reset.

CLKENS Clock enable for the slave port.

CLKENM Clock enable for the master port.

5.6.2 Functional interfaces

The ATB synchronous bridge consists of a slave ATB interface and a master ATB interface.

5.6.3 Operation

The ATB synchronous bridge has a master and a slave port. Both the ports operate on the same clock, **CLK**. The following describes the main functionality of the component:

Clock and reset

The bridge operates on a single clock, **CLK**. The clock enables are available at master and slave interfaces. This enables these interfaces to work at a lower frequency.

RESETn is used as an asynchronous reset throughout the entire design.

ATB data transfer slave interface

The assertion of **ATVALIDS** at the slave port initiates the ATB transfer start phase. If the internal buffer or FIFO is not FULL, the **ATREADYS** signal is immediately asserted to acknowledge the data capture, and the data is moved into the buffer. If the internal buffer is not empty, **ATREADYS** is held LOW to maintain the transfer status.

ATB data transfer master interface

When valid data is present in the buffer, the **ATVALIDM** signal is asserted to indicate that valid data is available on the master interface. The data is maintained until acknowledged by the assertion of **ATREADYM**. The **ATVALIDM** signal is held HIGH, until all the valid data is transmitted, and acknowledged.

Flush operation

The ATB flush is initiated on the master port by asserting the **AFVALIDM** signal. The flush request is propagated to the slave interface by asserting **AFVALIDS**. When the **AFREADYS** is driven HIGH, it indicates the completion of flush on the slave interface. When all the flush data is transferred to the Master interface from the internal buffers, **AFREADYM** is asserted indicating that the flush is complete on the master interface.

5.6.4 Syncreq

The ATB synchronous bridge propagates **SYNCREQ** from the ATB master port to the ATB slave port.

5.6.5 Low-power control

The ATB synchronous bridge is designed to be in a single power domain. The power domain boundary is outside this component, and the low-power interface provides the signal interface to the power controller.

The ATB synchronous bridge accepts low-power entry when it is requested from the system power controller.

The ATB synchronous bridge does not initiate wake-up. Exit from the low-power state is always initiated by the system controller.

The ATB synchronous bridge initiates ATB flush before acknowledging low-power entry to drain relevant data from the system.

When the flush is completed on the slave interface, the bridge acknowledges all transfers by asserting **ATREADYS** HIGH, but without storing them in the bridge.

The low-power state is entered only when the buffers inside the bridge do not have valid data.

Chapter 6
Timestamp Components

This chapter describes the timestamp component. It contains the following sections:

6.1 About the timestamp components

The timestamp components generate and distribute a consistent time value to multiple processors and other IP in a SoC. Figure 6-1 shows an example timestamp system.

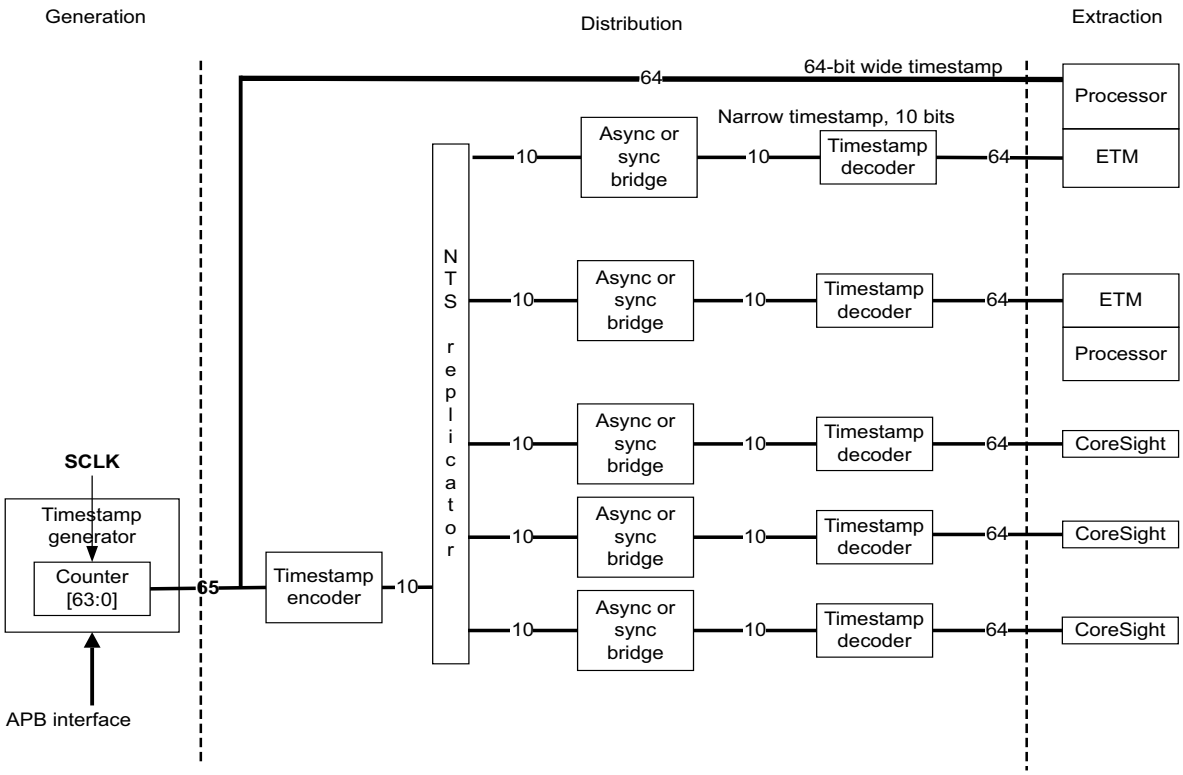


Figure 6-1 Timestamp example system

The timestamp interconnect provides a mechanism for efficiently distributing a timestamp value across a potentially large system in a way that is cost-effective to implement. It has the following features:

- Uses a master timing reference with a fixed frequency of typically 10-50 MHz.
- Time always counts forward.
- Time available as a natural binary number to software.
- Writeable and readable count value.
- Distributed synchronization of timestamp.
- Time value presented as 64-bit binary count.

The master timestamp generator must be programmed to match the actual clock rate. The interconnect ensures that any components that uses the distributed timestamp are synchronized to the distributed count value with minimal skew.

6.2 Timestamp solution

The timestamp solution includes the following individual components:

- *Timestamp generator*.
- *Timestamp encoder* on page 6-7.
- *Narrow timestamp replicator* on page 6-8.
- *Narrow timestamp asynchronous bridge* on page 6-8.
- *Narrow timestamp synchronous bridge* on page 6-10.
- *Timestamp decoder* on page 6-11.
- *Timestamp interpolator* on page 6-12.

Only the timestamp generator is programmable. The other components have no programmers model and operate autonomously.

See Chapter 2 *Functional Overview* for more information about block diagrams and key features.

6.2.1 Timestamp generator

The timestamp generator has the following features:

- Writeable or readable count value.
- Clocked by **SCLK**.

  SCLK This is the slow clock on which the timestamp generator operates. SCLK is the **clk** port on the timestamp generator.

  FCLK This is the fast local clock on which the timestamp interpolator operates. FCLK is the **clk** port on the timestamp interpolator.
- **SCLK** operates at a fixed frequency.
- Outputs time as a 64-bit binary number using the wide timestamp interface, with an indicator that the value has been written and must be resynchronized to all receivers.
- Increments at each tick of **SCLK**.
- It can be used for either one or both of the following:
  - CoreSight counter for timestamping.
  - ARM generic timer source.
- It can be optionally halted when any processor enters debug state.

This section describes the timestamp generator component and its functionality. It contains the following sections:

- *Clock and reset*.
- *Functionality* on page 6-4.
- *Register block* on page 6-4.
- *Counter* on page 6-6.

Clock and reset

The timestamp generator has the following clock and reset domain:

- A single clock, **CLK**.
- A reset, **RESETn**.

The reset, **RESETn**, must be asserted asynchronously, but deasserted synchronous to the clock, **CLK**.

Functionality

The counter has the following features:

- It runs at a constant clock frequency, regardless of the power and clocking state of the processor cores using it.
- It continues to run in all levels of powerdown other than completely turning off the system.
- Its size is 64-bit.
- It starts from 0.
- Its value can be read or written using 32-bit read on an APB interface.
- Its value can be written only when it is either halted or disabled.
- It has an APB interface. Access protections require assistance from other peripherals such as the *Trustzone Address Space Controller* (TZASC). See the *CoreLink TrustZone Address Space Controller TZC-380 Technical Reference Manual*.
- When the system is halted as a result of debug, the counter can be programmed to either halt or continue incrementing.

Register block

The register block has the following features:

- It implements all the programmable registers and status registers. See Chapter 3 *Programmers Model*.
- It implements APB interfaces, that is, a control interface and a *Read-Only* (RO) interface, in which all the inputs are registered using a single stage of flip flops.
- The RO interface is expected to map to a non-secure memory region and only supports read access to a limited subset of the register map.
- The control interface is expected to map to a secure memory region if the system supports it, and also supports read and write access.
  - It generates control signals to enable and disable counter operation.
  - It enables update of counter value based on a new programmed value.
- The control interface has a higher priority than the RO interface.
  - A fixed priority is implemented so that, if a read or a write request on this interface coincides with the read request on the RO interface, then requests on the RO have wait states while the control interface completes its transaction.
  - The read data path is registered. Therefore, all read accesses have at least one cycle of wait state.

  Note

  Because the control interface has higher priority, it always has only one cycle of wait state.

  At best, the RO interface has a single wait state, but if it coincides with an access on the control interface, then it has two wait states before the read access is complete.

Figure 6-2 shows the write access to enable counter and read access.

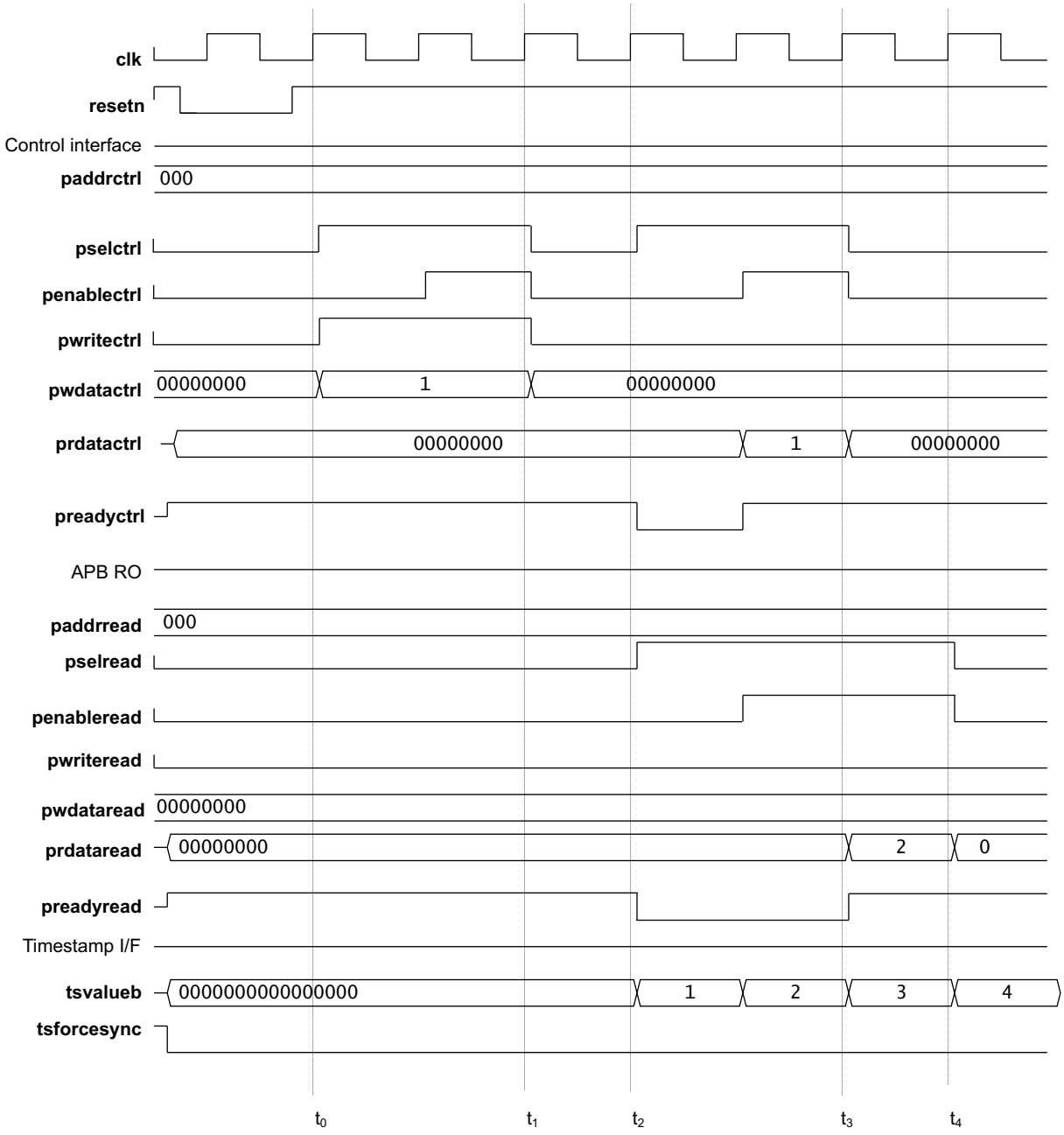


Figure 6-2 Write access to enable counter and read access

Table 6-1 shows event sequence of the write access to enable counter and read access.

Table 6-1 Event sequence of the write access to enable counter and read access

Time	Event
t_0	Start of a write access on the control interface to enable the counter.
t_1	The write access is completed and the counter is enabled.
t_2	Start of a read access on the control interface, that is, higher priority, and a simultaneous read access on the RO interface.
t_3	The access is completed on the control interface. The wait state is inserted on the RO interface by driving **preadyread** LOW
t_4	The access is completed on the RO interface.

- Two 32-bit writes are expected on the APB control interface before the 64-bit value can take effect.

  The writes must occur in the following sequence:

  1. The first write access must be to the lower 32 bits, the CNTVL Register.
  2. The second write access can be to the upper 32 bits, the CNTVU Register.
  3. The CNTVL Register can be updated any number of times before updating the CNTVU Register.

     Note

     The most recent value of CNTVL is retained.

  4. The CNTVU Register must be updated once for a 64-bit write to take effect.

- One of the following conditions must be met before the counter is updated with the new 64-bit value:
  - The counter must be disabled by resetting the EN bit in the CNTCR.
  - The counter must be halted by setting the HDBG bit in the CNTCR, and asserting the **HLTDBG** input.

If the condition is active during either the setup phase or the access phase of an APB write access to the upper 32 bits of the CNTVU Register, then the counter is halted and updated with the new 64-bit value.

Counter

The counter has the following features:

- The counter is implemented as two 32-bit incrementers.
- Update a new count value only when the counter is either disabled or halted. If the counter is still incrementing, then the request to update the count value is ignored.
- When a new count value is updated, **TSFORCESYNC** is asserted for one clock cycle to indicate that the **TSVALUEB** has a new count value.
- When the counter value rolls over from `64'hFFFF_FFFF_FFFF_FFFF` to `64'h0000_0000_0000_0000`, **TSFORCESYNC** is asserted for one clock cycle.

Figure 6-3 on page 6-7 shows the counter operation of the timestamp generator.

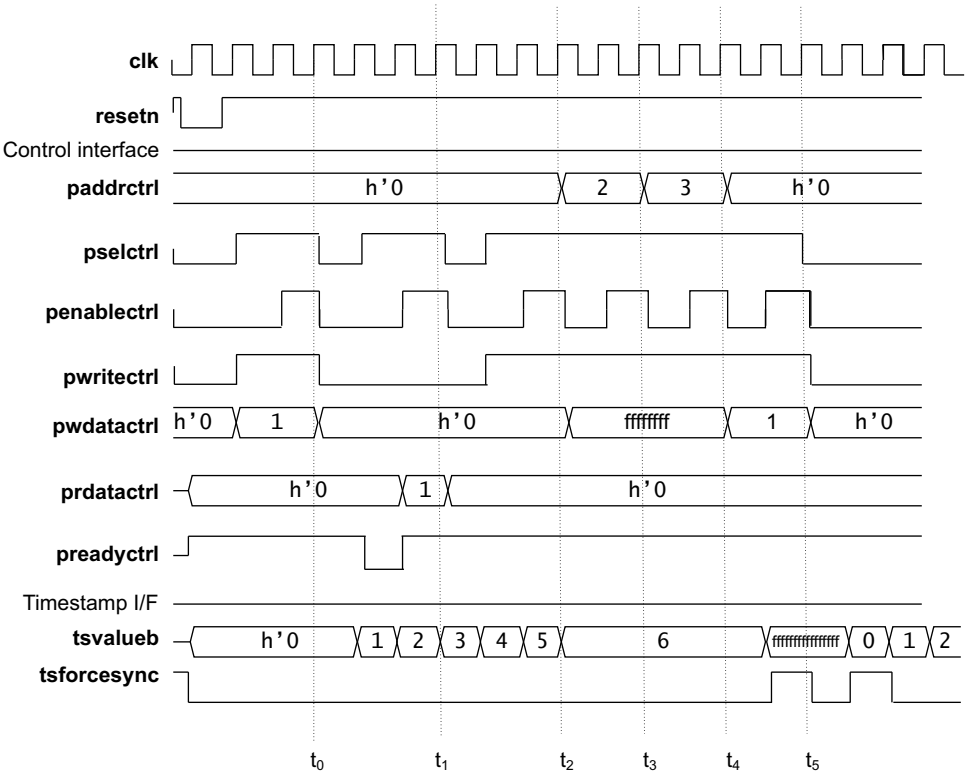


Figure 6-3 Counter operation

Table 6-2 shows the event sequence for the counter operation.

Table 6-2 Event sequence of the counter operation

Time	Event
t_0	• Write to the control register. • The counter is enabled.
t_1	Read from the control register.
t_2	• Write to the control register. • The counter is enabled.
t_3	Program the lower 32-bit of the count value.
t_4	• Program the upper 32-bit of the count value. • The **tsforcesync** is asserted for one clock cycle to indicate the update of a new count value.
t_5	• Write to the control register. • The counter is enabled. • The **tsforcesync** is asserted for one clock cycle is to indicate the roll-over.

6.2.2 Timestamp encoder

The timestamp encoder has the following features:

- Transforms a 64-bit binary count value to a 7-bit encoded value.

- Encodes and sends the binary count value over a 2-bit synchronization channel. Other CoreSight trace components use this information when they are powered up and must re-establish the latest timestamp.
- Works on a single clock input.

Synchronization information is continuously output over the 2-bit synchronization channel.

6.2.3 Narrow timestamp replicator

The narrow timestamp replicator is connected to the output of the encoder and has the following features:

- Distributes encoded data and the synchronization channel to multiple slave bridges.
- Can be connected to the output of an asynchronous or synchronous bridge for additional distribution to multiple bridges downstream. It can only be used to distribute to bridges in the same power domain.
- Combines the ready input from multiple slave bridges and provides a unified ready output to the master encoder or bridge.
- When it is connected to encoder, tie `ts_bit_valid_qualify` as LOW and when it is connected to an asynchronous or synchronous bridge, tie `ts_bit_valid_qualify` as HIGH.

6.2.4 Narrow timestamp asynchronous bridge

The narrow timestamp asynchronous bridge has two clock domains and a reset input for each of the clock domains. Resets are expected to be asserted asynchronously but deasserted synchronously with respect to the respective clock domains.

This acts as a bridge between two different clock and power domains. This component primarily has an asynchronous FIFO and an LPI responding to low-power requests and transferring data between different clock domains. It also performs additional operations specific to the timestamp interconnect. It has an additional storage that has a depth of one.

When the FIFO is full and cannot accept any more data, this component begins to store the next highest **TSBIT** value that it receives in the additional storage space. It continuously updates the next highest **TSBIT** value replacing it with the value it receives in the current cycle, if this value is greater than the value it stored previously.

On the LPI interface, the **CACTIVE** is always driven LOW.

When a powerdown request is asserted on the LPI interface:

- **CSYSREQ** stops any additional values being written to the FIFO.
- **CSYSACK** is asserted when the master interface acknowledges that it has reset its read pointer by sending an acknowledgement.
- The master interface empties the FIFO contents after which the read pointer is reset.
- The writer pointer in the slave interface is now reset.
- **TSSYNCREADYS** is driven HIGH when **CSYSREQ** is LOW.
- **TSSYNCM** is driven to `0b00` to indicate an invalid value, when **CSYSACK** is driven LOW.
- **TSBITM** is also reset when CSYSACK is driven LOW.

On deassertion of a powerdown request, and after it enters the power-up state, it sends a resync packet, 7', on **TSBITM** so that all the slaves downstream can resynchronize to the new count value.

You must ensure that the cumulative depth of the cascaded asynchronous FIFO does not exceed the number of **TSSYNC** packets required to send one complete synchronization sequence.

Figure 6-4 shows the low-power entry and exit timing diagram.

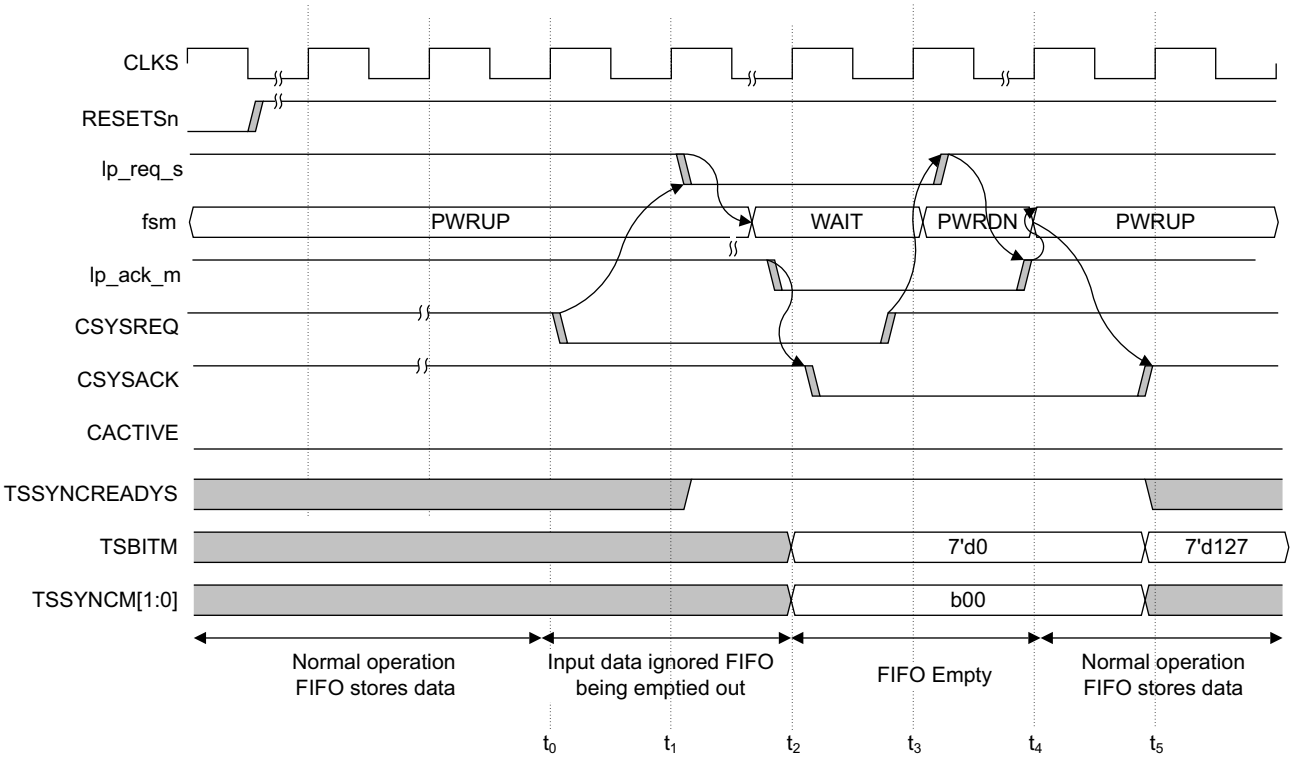


Figure 6-4 Low-power entry and exit timing diagram

Table 6-3 shows event sequence of the low-power entry and exit timing diagram of the narrow timestamp asynchronous bridge.

Table 6-3 Event sequence of the low-power entry and exit timing diagram

Time	Event
t_0	• The **CSYSREQ** is driven LOW. • The powerdown request is detected.
t_1	The powerdown request sent to the master interface.
t_2	• The powerdown acknowledgement is received from the master interface. • The **CSYSACK** is driven LOW, to indicated the completion of powerdown.
t_3	• The **CSYSREQ** is driven HIGH. • The powerup request is detected. • The powerup request is sent to the master interface.
t_4	The powerup acknowledgement is received from the master interface.
t_5	The **CSYSACK** is driven HIGH, to indicate the completion of powerup.

6.2.5 Narrow timestamp synchronous bridge

The narrow timestamp synchronous bridge has a single clock domain and a reset input. Reset is expected to be asserted asynchronously but deasserted synchronously. This bridge has two clock-enables, one for the master interface and the other for the trace interface.

This bridge primarily has a register stage and an LPI responding to low-power requests and transferring data based on individual clock enables. When the slave module is unable to read new data from this bridge, it begins to store the next highest TSBIT value that it receives.

If the value received in the current cycle is greater than the value previously stored, it updates the next highest TSBIT value, replacing it with the new value.

On the LPI interface, the **CACTIVE** is always tied LOW.

When a powerdown request is asserted on the LPI interface:

- **CSYSREQ** stops any additional values being stored.
- **CSYSACK** is asserted.
- **TSSYNCREADYS** is driven HIGH when **CSYSREQ** is LOW.
- **TSSYNCM** is driven to `0b00` to indicate an invalid value, when **CSYSACK** is driven LOW.
- **TSBITM** is also reset when **CSYSACK** is tied LOW.

On deassertion of a powerdown request, and after it enters the power-up state, it sends a resync packet, `7'd127`, on **TSBITM** so that all the slaves downstream can resynchronize to the new count value.

You must ensure that the cumulative depth of all the register stages does not exceed the number of **TSSYNC** packets required to send one complete synchronization sequence.

Figure 6-5 on page 6-11 shows the low-power entry and exit timing diagram of the narrow timestamp synchronous bridge.

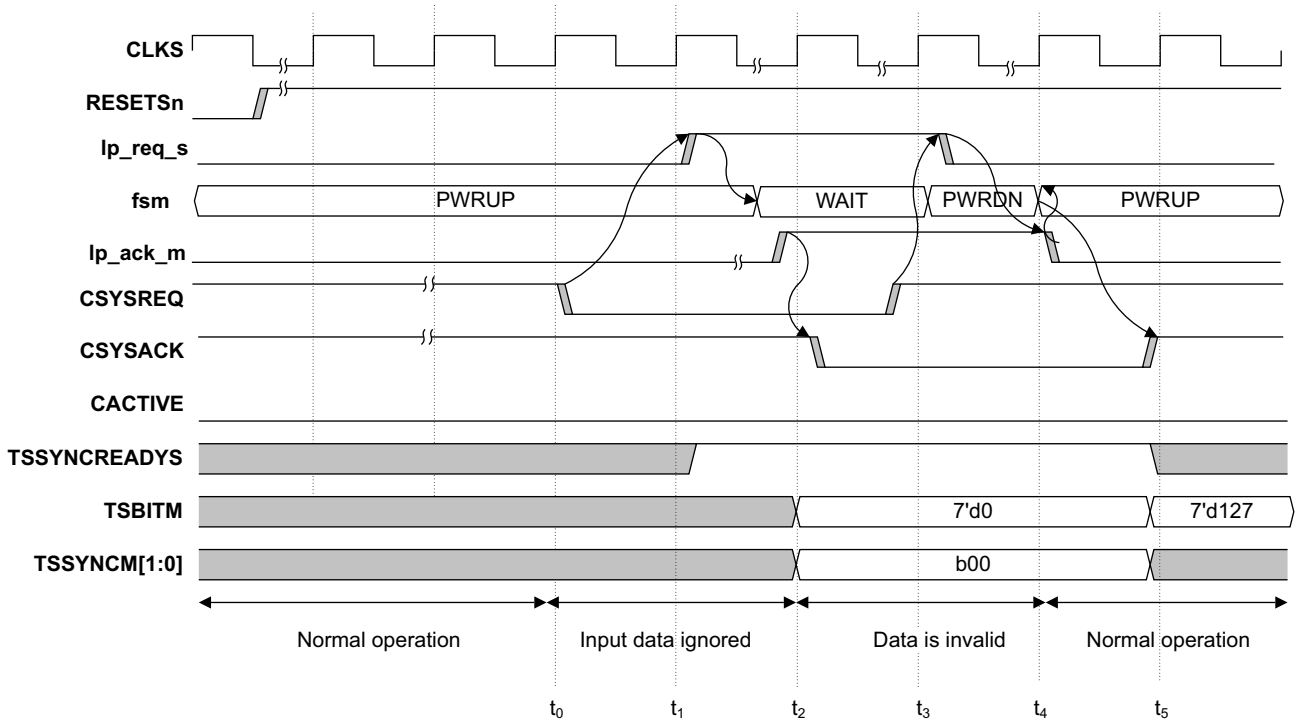


Figure 6-5 Low-power entry and exit timing diagram

Table 6-4 shows event sequence of the low-power entry and exit timing diagram of the synchronous bridge.

Table 6-4 Event sequence of the low-power entry and exit timing diagram

Time	Event
t_0	• The **CSYSREQ** is driven LOW. • The powerdown request is detected.
t_1	The powerdown request is sent to the master interface.
t_2	• The powerdown acknowledgement is received from the master interface. • The **CSYSACK** is driven LOW, to indicated the completion of powerdown.
t_3	• The **CSYSREQ** is driven HIGH. • The powerup request is detected. • The powerup request is sent to the master interface.
t_4	The powerup acknowledgement is received from the master interface.
t_5	The **CSYSACK** is driven HIGH, to indicate the completion of powerup.

6.2.6 Timestamp decoder

The timestamp decoder decodes the 7-bit encoded value to generate a 64-bit binary count value and:

- Uses this information from a CoreSight trace component that is powered up and requires the latest timestamp value.

- Decodes the timestamp value in the following steps:
  - Reconstructs the latest timestamp value by using information from the synchronization channel.
  - Increments the count value based on the encoded value.

When the timestamp decoder reconstructs the latest timestamp value, it drives **tsvalue[63:0]** to 0 to indicate that the timestamp value is invalid and it is in the process of reconstructing a new timestamp value.

When the timestamp decoder detects a value of `0x7F` on the tsbit, it stops incrementing the count value and starts monitoring the synchronization channel to reconstruct the new timestamp.

6.2.7 Timestamp interpolator

The Timestamp Interpolator takes in 64-bit timestamp values that change at every system clock, **SCLK** tick. It outputs a modified count related to a second clock source **FCLK**, which is usually faster than **SCLK**.

Clock and reset

This component has a single clock domain, **clk**.

This component has a single reset input, **resetn**, which is an asynchronous active-LOW reset input.

Functional interface

This component has:

- A wide timestamp input interface that carries the un-interpolated timestamp values from the timestamp generator.
- A wide timestamp output interface that carries the interpolated timestamp values.

Functional description

The following are some of the key characteristics of the timestamp interpolator:

- The timestamp interpolator takes in 64-bit timestamp values that increment at every **SCLK** tick. **FCLK** is usually faster than **SCLK**:

  SCLK This is the slow clock on which the timestamp generator operates. **SCLK** is the **clk** port on the timestamp generator.

  FCLK This is the fast local clock on which the timestamp interpolator operates. **FCLK** is the **clk** port on the timestamp interpolator.

- The interpolator discards the upper N bits, appends N bits from its internal incrementing logic, and sends out **tsvalueintpb[63:0]**. N is the number of interpolated timestamp bits that the system designer configures.
- If the incoming timestamp value is 0, the timestamp interpolator drives the output value to 0.
- The interpolated value changes on every rising edge of **FCLK**. The interpolated bits increment to ensure that they reach the maximum value when the higher order bits change in the incoming timestamp.

- The timestamp interpolator resets the interpolated bits for every change detected in the incoming timestamp.
- To enhance the linear accuracy of interpolation, the timestamp interpolator interpolates additional bits, M, but only uses the upper N-bits of the result. M is defined as 4 and it cannot be changed.
- The timestamp interpolator ensures that the output timestamp value does not decrease, provided that the input timestamp value does not decrease with respect to its previous timestamp value.
- At slower **SCLK** frequencies, the timestamp generator counts in higher increments to compensate and maintain a constant speed relative to real time. In this scenario, the interpolator interpolates up to P additional bits. At run-time, the number of extra bits interpolated is Q, in the range 0 to P.
- The Timestamp interpolator locates the least-significant bit in the Timestamp generator value that changes to calculate Q. When **SCLK** is at its maximum frequency, Q is 0.

Note

The timestamp interpolator must not be part of the generic time distribution network.

The following characteristics apply to are Interpolator behavior under special conditions:

- If the **tsvalueb** input is 0, the timestamp interpolator drives 0 on **tsvalueintpb** output and waits for the next increment on **tsvalueb** input.
- The incrementer saturates at the maximum value, when both of the following conditions apply:
  - The incrementer reaches the maximum value for the calculated value of Q.
  - The next timestamp value is not available on **tsvalueb**.
- When the input clock ratio exceeds the value that you configured, the interpolator still operates at the configured value of clock ratio.
- When **SCLK** is faster than **FCLK**, the interpolator does not interpolate and the **tsvalueb** value sampled on the input is driven on the tsvalueintpb output. Some values on the **tsvalueb** input might be lost on the **tsvalueintpb** output because of the difference in the frequencies.

Chapter 7
Embedded Cross Trigger

This chapter describes the ECT, CTI, and the CTM. It contains the following sections:

7.1 About the ECT

The ECT for CoreSight consists of a number of CTIs and CTMs connected together. You can operate a single CTI without the requirement for a CTM.

7.1.1 How ECT works

The ECT provides an interface to the debug system as Figure 7-1 shows. This enables ARM/ETM subsystems to interact, that is, cross trigger with each other. The debug system enables debug support for multiple cores, together with cross triggering between the cores and their respective ETMs.

The main function of the ECT is to pass debug events from one core to another. For example, the ECT can communicate debug state information from one core to another, so that program execution on both processors can be stopped at the same time, if required.

Cross Trigger Interface

The CTI combines and maps the trigger requests, and broadcasts them to all other interfaces on the ECT as channel events. When the CTI receives a channel event it maps this onto a trigger output. This enables subsystems to cross trigger with each other. The receiving and transmitting of triggers is performed through the trigger interface.

Cross Trigger Matrix

This block controls the distribution of channel events. It provides *Channel Interfaces* (CIs) for connection to either CTIs or CTMs. This enables multiple CTIs to be linked together.

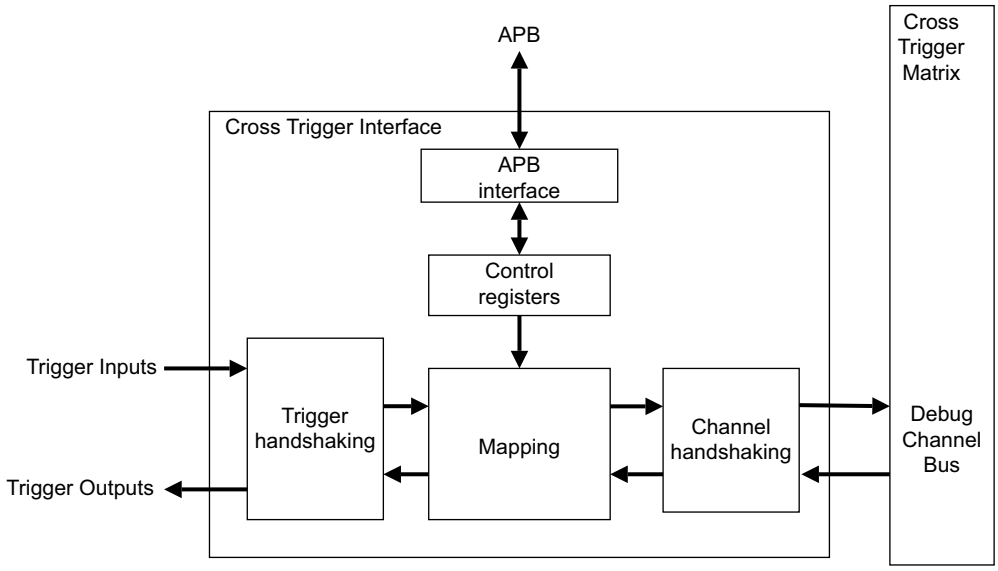


Figure 7-1 CoreSight CTI and CTM block diagram

7.1.2 CTI handshaking, synchronization, and clocks

This section describes handshaking, synchronization, and clocks. It contains the following sections:

- *Interfaces handshake protocol* on page 7-3.
- *Synchronization* on page 7-3.
- *Clock domains* on page 7-4.

Interfaces handshake protocol

The CTI does not interpret the signals on the CI or *Trigger Interface* (TI). If handshaking is enabled, then it is assumed the events are edge-triggered. As a result, closely occurring events cannot reliably be recognized. An event is transmitted as a level. If you require edge detection or single pulse output you must implement the necessary shaping logic in the external wrapper.

To avoid any incompatibility, the following protocols are defined:

- Only logic 1 is interpreted as an event.
- If handshaking is enabled, an output must stay active until an acknowledgement by hardware or optionally acknowledged by software for the TI is received, even if the acknowledge signal driver is deactivated.

If handshaking is enabled, the CTI can only handle one-shot events. If events are close to one another, or multi-shot, and mapped to the same channel, they are possibly merged into one event on an output trigger. For debug events such as breakpoint, trace start, and trace stop this does not cause a problem because input events mapped to the same triggers, must do the same thing.

Events arising from different interfaces but mapped to the same channel might also be merged. This is acceptable because the mapping logic can be programmed to enable this. Events can be merged because block handshakes between asynchronous clock domains or channels events are mapped onto the same output trigger.

If the events sent on the CTI emanate from the same clock domain then you can bypass the handshaking and synchronizing logic. The output does not receive an acknowledgement. In such cases you can use the CTI to transmit multiple shot events. The output has to remain active for at least one clock cycle to ensure that it is captured at the destination interface.

Synchronization

Some systems might require clock-domain crossing synchronizers for inputs and outputs to the CTI. This can result in variations in the delay that can be expected between a signal that is generated and the signal that is sampled.

Channel interface handshaking

The channel interface handshaking used for this asynchronous interface is a basic four-phase handshake protocol. See Figure 7-2.

The same protocol is used by **ctichout** and **ctichoutack**.

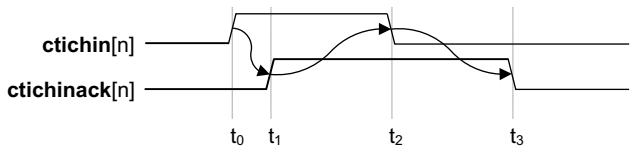


Figure 7-2 Channel interface handshaking

Table 7-1 shows event sequence of the channel interface handshaking.

Table 7-1 Event sequence of the channel interface handshaking

Time	Event
t_0	The **ctichin[n]** is asserted, but it is not acknowledged.
t_1	The **ctichin[n]** is asserted and acknowledged.
t_2	The **ctichin[n]** is de-asserted, but it is not acknowledged.
t_3	The **ctichin[n]** is de-asserted and acknowledged.

See the *CoreSight Architecture Specification*.

Clock domains

In most systems, the **cticlk** is connected to the same clock as its local processor and the **ctmclk** is connected to the fastest processor clock in the system. This reduces the trigger latency and the requirement for clock enable ports.

If a CTI is disabled while the processor enters a clock stopped mode, ARM recommends that:

- The CTI firstly turns off the event-to-channel mapping, so that unwanted events are not generated to the CTM.
- The channel-to-event mapping circuit is turned off or disabled for a few clock cycles after the clock is on.
- This version of the CTI must not be connected to clocks that might be removed, that is stopped. This version of the CTI must not be placed within a power domain that can be turned off. In these scenarios, ARM recommends that **cticlk** is the same as **ctmclk**.

When a processor clock is stopped, for example, waiting for an interrupt, the corresponding CTI can receive an event from the CTM. When the CTI clock is the same as the subsystem clock and the handshaking is not bypassed, the CTM keeps the signal active until an acknowledgement is received, which only occurs when the clock is started again. In this case, out-of-date events can happen on the core. This does not inhibit the channel being used by other processors.

However, if the CTI clock differs from the local processor clock, for example, it is gated differently, it is possible for the CTI to raise an event to the core using **ctitrigout**, while the processor clock is off. If raising an event to the core must be avoided, the processor must disable its CTI before stopping the clock.

Linking CTIs and CTMs

Where the clock used on a CTI and a connected CTM, or on a CTM and a connected CTM, or on a CTI and a connected CTI, is asynchronous, then both the handshaking logic and synchronization registers must be left enabled, that is, **cihsbypass** and **cisbypass** must be tied LOW.

If both devices have synchronous clocks then synchronization can be bypassed and **cisbypass** tied HIGH, to reduce latency.

If both clocks are the same, that is, **ctmclk** = **cticlk**, and the channel is required to send multi-shot events, the handshaking can be bypassed by tying **cihsbypass** HIGH.

7.2 ECT programmers model

The base addresses of the CTIs are not fixed, and can be different for any particular system implementation. However, the offset of any particular register from the base address is fixed. Each CTI has a 4KB programmers model. Each CTI must be programmed separately. All unused memory space is reserved.

The following applies to all registers:

- Reserved or unused bits of registers must be written as 0, and ignored on a read unless otherwise stated in the text.
- All register bits are reset to 0 unless otherwise stated in the text.
- All registers must be accessed as words and so are compatible with big-endian and little-endian systems.

Note

The CTM does not have a programmers model because it is a link between two or more CTIs or additional CTMs.

For more information, see Chapter 3 *Programmers Model*.

7.3 ECT connectivity recommendations

Contact ARM for full information on the standardized connections between ARM processors, CoreSight components, and the ECT interconnect.

7.4 ECT authentication requirements

This section describes the functionality that must be available in the ECT to permit authentication using the signals described in the *Additional reading* on page ix, and describes how they must be connected. If a system does not support this level of control, then simplifications of the system design can be made.

The device does not require any inputs that are capable of disabling it as a whole. While it is possible for the device to be invasive, by asserting interrupts, it must continue to function when **dbgen** is LOW, because it might be required for non-invasive debugging, for example to communicate profiling events or a trigger condition. As a result, individual trigger inputs and outputs must be masked as required.

7.4.1 Trigger inputs

The trigger inputs must be masked by **niden** where they are not connected to another debug or trace device.

7.4.2 Trigger outputs

The trigger outputs must be masked by **dbgen** where they might otherwise affect the behavior of a running system and do not first require to be specifically enabled by the system. Table 7-2 shows the signals that ARM recommends. See *Additional reading* on page ix.

Table 7-2 ECT recommended trigger outputs

Destination signal	Destination device	Masking required	Comments
DBGRQ, **EDBGRQ**	Core	No	This signal is ignored when **DBGEN** is LOW, and requires no more masking.
VICINTSOURCE	VIC	No	Privileged system software must explicitly enable the interrupt source for this signal to have any effect.
nIRQ[a]	Core	Yes	Directly changes the execution flow of the core.
EXTIN	ETM	No	Only affects the operation of the ETM.
DBGEXT, EXTERN0, **EXTERN1**	Core	No	Only affects the operation of the debug logic.
ETMEXTOUT	Core	No	Only affects the operation of the PMU.

a. Inverted.

7.4.3 ECT authentication signals

Table 7-3 shows the required authentication signals.

Table 7-3 ECT authentication signals

Signal	Direction	Description
dbgen	Input	Debug enable signal to mask trigger outputs from a CTI.
niden	Input	Non-invasive debug enable input to mask the trigger inputs that are not connected to a trace or debug device.
tinidenselx[7:0]	Input	Select to enable individual trigger inputs to be masked when **niden** is LOW. Each bit of **tinidenselx[7:0**] must be asserted HIGH to bypass **niden**, or LOW to be masked by **niden**.
todbgenselx[7:0]	Input	Select to enable individual trigger outputs to be masked when **dbgen** is LOW. Each bit of **todbgenselx[7:0]** must be asserted HIGH to bypass **dbgen**, or LOW to be masked by **dbgen**.

If **niden** is LOW and **tinidenselx** is LOW, then any read of the CTI Trigger In Status Register returns the corresponding bit LOW. This applies to both normal operation mode and integration test mode. If the CTI is configured to generate trigger acknowledgements this must be maintained in normal mode, irrespective of the state of **niden**.

If **dbgen** is LOW and **todbgenselx** is LOW, then any read of the CTI Trigger Out Status Register register returns the corresponding bit LOW. The CTTRIGOUTx register is also LOW. This applies to normal operation mode. In Integration test mode all writes to the ITTRIGOUT Register are ignored. The CTTRIGOUTx register is LOW.

Chapter 8
Trace Port Interface Unit

This chapter describes the TPIU. It contains the following sections:

8.1 About the Trace Port Interface Unit

The TPIU acts as a bridge between the on-chip trace data, with separate IDs, and data stream, encapsulating IDs where required, that is then captured by a *Trace Port Analyzer* (TPA). Figure 8-1 shows the main blocks of the TPIU.

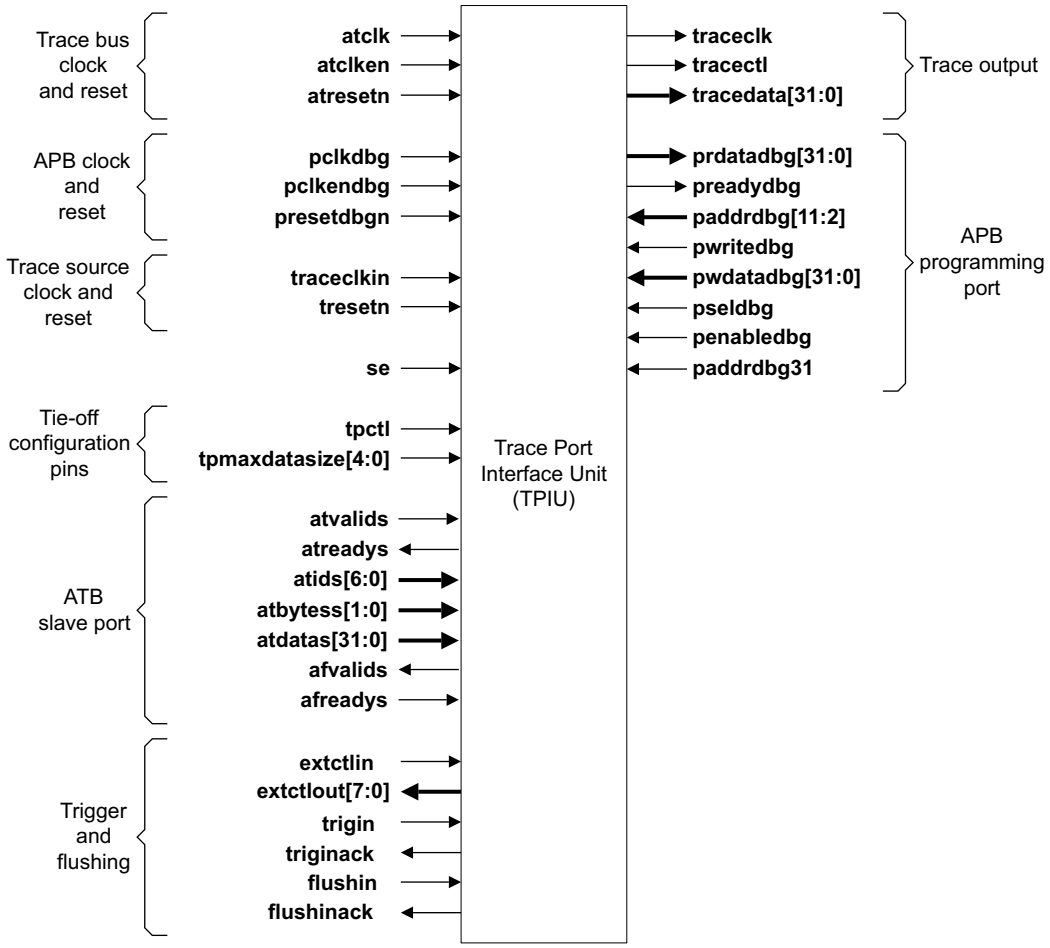


Figure 8-1 TPIU block diagram

The behavior of the blocks is as follows:

Formatter Inserts source ID signals into the data packet stream so that trace data can be re-associated with its trace source. See *TPIU formatter and FIFO* on page 8-14.

Register bank

Contains the management, control, and status registers for triggers, flushing behavior, and external control.

Trace out The trace out block serializes formatted data before it goes off-chip.

Pattern Generator

The pattern generator unit provides a simple set of defined bit sequences or patterns that can be output over the Trace Port and be detected by the TPA or other associated *Trace Capture Device* (TCD). The TCD can use these patterns to indicate whether it is possible to increase or to decrease the trace port clock speed. See *TPIU pattern generator* on page 8-12.

8.1.1 ATB interface

The TPIU accepts trace data from a trace source, either direct from a trace source or using a trace funnel.

8.1.2 APB interface

The APB interface is the programming interface for the TPIU.

8.2 Trace Out Port

Table 8-1 shows the Trace Out Port signals. For more information about TPIU registers, see Chapter 3 *Programmers Model*.

Table 8-1 Trace Out Port signals

Name	Type	Description
traceclkin	Input	Decoupled clock from ATB to enable easy control of the trace port speed. This is typically derived from a controllable clock source on chip but can be driven by an external clock generator if a high speed pin is used. Data changes on the rising edge only. See *Off-chip based traceclkin* on page 8-9 for more information about off-chip operated **traceclkin**.
traceclk	Output	Exported version of **traceclkin**. This is **traceclkin**/2, and data changes on both rising edges and falling edges. See *traceclk generation* on page 8-9 for more information about **traceclk** generation.
tracedata[mps:0]	Output	Output data. MPS is **tpmaxdatasize**.
tracectl	Output	Used to indicate non-valid trace data and triggers. See *Other TPIU design considerations* on page 8-9.
tresetn	Input	This is a reset signal for the **traceclkin** domain. Because off-chip devices connect to the Trace Out port, this signal is related to the Trace Bus Resetting signal, **atresetn**.
tpctl	Input	ASIC tie-off to report the presence of the **tracectl** pin. If **tracectl** is not present then this must be tied LOW. This input affects bit 2 of the Formatter and Flush Status Register.
tpmaxdatasize[4:0]	Input	Tie-off to indicate the maximum **tracedata** width available on the ASIC. The valid values are 1-32 (`0x00-0x1F`). For example, if only a maximum of a 16-bit data port is available, this takes the value `0x0F`. This input affects the Supported Port Size Register.

8.3 Miscellaneous connections

These ports supplement the operation of the TPIU, and are for the connection of other CoreSight components or other ASIC blocks. Table 8-2 shows ports that are not described elsewhere. For more information about TPIU registers, see Chapter 3 *Programmers Model*

Table 8-2 TPIU miscellaneous ports

Name	Type	Description
trigin	Input	From either a CTI or direct from a trace source. Used to enable the trigger to affect the output trace stream.
triginack	Output	Return response to the CTI acknowledgement to **trigin**.
flushin	Input	External control used to invoke an ATB signal and drain any old historical information on the bus.
flushinack	Output	Flush response, goes HIGH to acknowledge **flushin**. If the completion of a flush is required, then this can be established by the FFCR.
extctlout[7:0]	Output	Used as control signals for any configurable drivers on the output of the trace port, such as serializers and output multiplexer.
extctlin[7:0]	Input	Used as an input port for any configurable drivers on the output of the trace port, such as serializers and output multiplexer.

8.4 TPIU trace port sizes

The TPIU is configured for the largest port size permissible, 32-bit of **tracedata**, **traceclk**, and **tracectl**.

- **traceclk** is always exported to enable synchronization back with the data and so is not optional.
- **tracectl** is required unless a new TPA is used that is aware of the formatter protocol which can remove extra packets used to expand data sequences. For normal and bypass modes, **tracectl** must be present.
- **tracedata** can be defined as any size up to 32-bit. For backwards compatibility and usage with ETMv3 trace capture devices, a minimum port width of 2-bit is permitted, that is, **tracedata[1:0]**.

Table 8-3 shows some typical Trace Out Port sizes.

Table 8-3 Example Trace Out Port sizes

traceclk present	tracectl present	tracedata width	Total pin count	Comment
Yes	Yes	32-bit, [31:0]	34	Largest implementation.
Yes	No	9-bit, [8:0]	10	Extra data pin available in comparison to the typical ETM implementation.
Yes	Yes	8-bit, [7:0]	10	Typical ETM-compatible TPA implementation.
Yes	Yes	- bit, [1:0]	4	Smallest implementation with typical TPAs.
Yes	No	1-bit, [0]	2	Smallest implementation with a protocol-aware TPA.

8.4.1 Programming registers

There are two registers that contain information relating to the physical Trace Out port. These are the Supported Port Size Register and the Formatter and Flush Status Register. For more information about these registers, see Chapter 3 *Programmers Model*.

Constraining the supported tracedata port widths

If the **tracedata** width is less than the number of data bits available, then the least significant bits are connected to **tracedata**. The **tpmaxdatasize[4:0]** indicates the number of valid bits of **tracedata.**

The tie-off must be set to represent the number of **tracedata** connections that go to ASIC pads. All LOW indicates 1 pin, which is the minimum possible, and all HIGH indicates 32 pins, which is a full **tracedata** bus. For example, if a 16-bit trace port is implemented, that is **tracedata[15:0]** is connected, then **tpmaxdatasize[4:0]** must be tied to `0x0F`. With a maximum **tracedata[7:0]** connected to the ASIC, the tie-offs must be set to `0x07`.

8.4.2 Omission of tracectl

For restricted pin devices where removal of the trace data is important, the **tracectl** can be removed. Omitting **tracectl** is only possible when the formatter is enabled and continuous mode is selected. For more information about the FFCR, see Chapter 3 *Programmers Model*.

The presence of the **tracectl** pin can be checked by reading the TCPresent bit in the FFSR. See *Formatter and Flush Status Register* on page 3-157.

8.5 TPIU triggers

Currently the only usage of triggers is by the trace capture device. This method is straightforward when using one trace source. When using multiple trace sources there can be a time disparity between the trace sources that generate a trigger and when the trigger packet appears at the output of the trace port. See the *CoreSight Architecture Specification* for more information on triggers.

A trigger can be interpreted as an event that occurred. This can be:

- Directly from an event such as a pin toggle from the CTI.
- A delayed event such as a pin toggle that has been delayed coming through the Trigger Counter Register.
- The completion of a flush.

Table 8-4 extends the ETMv3 specification on how a trigger is represented

Table 8-4 CoreSight representation of triggers

tracectl	tracedata		Trigger	Capture	Description
	[1]	[0]	Yes/No	Yes/No	
0	x	x	No	Yes	Normal trace data
1	0	0	Yes	Yes	Trigger packet[a]
1	1	0	Yes	No	Trigger
1	x	1	No	No	Trace disable

a. The trigger packet encoding is required for the current ETMv3 protocol that uses a special encoding for triggers that always occur on the lower bits of **tracedata**.

8.5.1 Correlation with afvalid

When the TPIU receives a trigger signal, depending on the Formatter Control Register, the **afvalid** port can be asserted to cause a flush of all current trace information. This causes all information around the trigger event to be flushed from the system before normal trace information is resumed. This ensures that all information related to the trigger is output before the TPA, or other capture device, is stopped.

With **FOnTrig** HIGH, it is possible to indicate the trigger on completion of the flush routine. This ensures that if the TPA stopped the capture on a trigger, the TPA gets all historical data relating to the trigger. For more information about FFCR, See Chapter 3 *Programmers Model*.

8.6 Other TPIU design considerations

This section describes TPIU design considerations in:

- *traceclk generation*.
- *tracectl removal*.
- *tracectl and tracedata multiplexing*.
- *Off-chip based traceclkin*.

8.6.1 traceclk generation

At implementation time, a **traceclk** output must be generated as a sampling reference for **tracedata**. Frequently, this is specified to fall mid-way between **tracedata** transitions. However for optimum performance a TPA can implement variable delays for sampling each individual **tracedata** signal.

8.6.2 tracectl removal

The TPIU supports two modes:

- data + control + clock, with a minimum data width of 2.
- data + clock, with a minimum data width of 1.

The chosen mode depends on the connected trace port analyzer or capture device. Legacy capture devices use the control pin to indicate the packet type. Newer capture devices can use more pins for data and do not require a reserved data pin.

Support for both of these modes is required to ensure backwards compatibility and for future, higher port speeds. If a low pin count or an optimized design is required, it is not necessary to implement the **tracectl** pin. This design choice must be reflected in the programmers model to enable tools to always enable the formatter and run in continuous mode.

8.6.3 tracectl and tracedata multiplexing

If pin minimization is a priority, and it is also necessary to support legacy TPAs that still require **tracectl**, it is possible to support both systems in the same implementation by multiplexing the **tracectl** pin with a data pin. This enables the support of the current method of using the control pin at the same time as enabling the connection of a next generation trace capture device with the added advantage of an extra data pin. A possible choice for this extra data pin is the most significant bit because this can be switched without having to change the signal paths of any other connection.

The ability to switch **tracectl** with a data pin is not directly supported by the TPIU. Problems can arise when trying to drive multiple connector pins for connector reuse, because of impedance and load differences. In addition, timing can be affected because of the inclusion of a multiplexer on a limited number of signals

8.6.4 Off-chip based traceclkin

Future CoreSight-aware TPAs might directly control a clock source for the Trace Out port. By running through a known sequence of patterns, from the pattern generator within the TPIU, a TPA can automatically establish the port width and ramp up the clock speed until the patterns degrade, thereby establishing a maximum data rate. Figure 8-2 on page 8-10 shows how to generate an off-chip **traceclkin**.

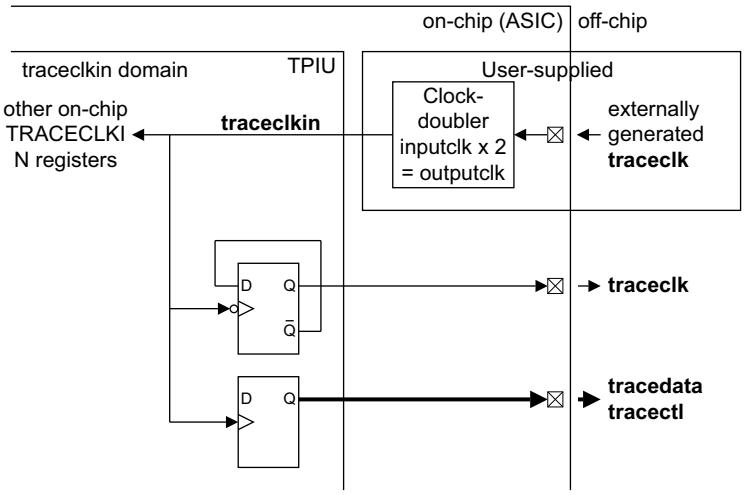


Figure 8-2 Externally derived traceclk

The off-chip clock can operate in a similar way to the currently exported **traceclk**. For example, an externally derived clock source can be double clocked to enable the exported data to change at both edges of the clock.

8.7 Authentication requirements for TPIUs

TPIUs do not require any authentication signals capable of disabling them.

8.8 TPIU pattern generator

A simple set of defined bit sequences or patterns can be output over the trace port and be detected by the TPA or other associated trace capture device. Analysis of the output can indicate whether it was possible to increase or, for reliability, to decrease the trace port clock speed. The patterns can also be used to determine the timing characteristics and so alter any delay components on the data channels in a TCD, to ensure reliable data capture.

8.8.1 Pattern generator modes of operation

There are a number of supported patterns to enable a number of metrics to be determined, for example, timing between pins, data edge timing, voltage fluctuations, ground bounce, and cross talk. When examining the trace port you can choose from the following pattern modes:

Timed Each pattern runs for a programmable number of **traceclkin** cycles after which the pattern generator unit reverts back to an off state where normal trace is output, assuming trace output is enabled. The first thing the trace port outputs after returning to normal trace is a synchronization packet. This is useful with special trace port analyzers and capture devices that are aware of the pattern generator. The TPIU can be set to a standard configuration that the capture device expects. The preset test pattern can then be run. By the end of which the TCD is calibrated ready for normal operation. The TPIU switches to automatically, without the requirement to reprogram the TPIU.

Continuous The selected pattern runs continuously until manually disabled. This is primarily intended for manual refinement of electrical characteristics and timing.

Off When neither of the other two modes is selected, the device reverts to outputting any trace data. After timed operation finishes, the pattern generator reverts back to the off mode.

8.8.2 Supported options

Patterns operate over all the **tracedata** pins for a given port width setting. Test patterns are aware of port sizes and always align to **tracedata[0]**. Walking bit patterns wrap at the highest data pin for the selected port width even if the device has a larger port width available. Also, the alternating patterns do not affect disabled data pins on smaller trace port sizes.

Walking 1s

All output pins clear with a single bit set at a time, tracking across every **tracedata** output pin. This can be used to watch for data edge timing, or synchronization, high voltage level of logic 1, and cross talk against adjacent wires. Walking 1s can also be used as a simple way to test for broken or faulty cables and data signals.

Walking 0s

All output pins are set with a single bit cleared at a time, tracking across every **tracedata** output pin. In a similar way to the walking 1s, walking 0s can be used to watch for data edge timing, or synchronization, low voltage level of logic 0, cross talk, and ground lift.

Alternating AA/55 pattern

Alternate **tracedata** pins are set with the others clear. This alternates every cycle with the sequence starting with **tracedata[1]** set to AA pattern = `0b1010_1010`, followed by **tracedata[0]** set to 55 pattern = `0b0101_0101`. The pattern repeats over the entire selected bus width. This pattern can be used to check voltage levels, cross talk, and data edge timing.

Alternating FF/00 pattern

On each clock cycle, the **tracedata** pins are either all set FF pattern or all cleared 00 pattern. This sequence of alternating the entire set of data pins is a good way to check the power supply stability to the TPIU and the final pads because of the stresses the drivers are under.

Combinations of patterns

Each selected pattern is repeated for a defined number of cycles before moving onto the next pattern. After all of the patterns are performed, the unit switches to normal tracing data mode. If some combination is chosen and the continuous mode is selected, each pattern runs for the number of cycles indicated in the repeat counter register before looping around enabled parameters.

8.9 TPIU formatter and FIFO

This section describes the functionality of the formatter and the FIFO.

The formatter is the final unit on the ATB that inserts the **atid[6:0]** into a special format data packet stream to enable trace data to be reassociated with a trace source.

The FIFO is 32-bit wide. To reduce the amount of logic required for this operation and that of the high-speed bridge, the FIFO is asynchronous.

8.9.1 Operational description

Figure 8-3 shows the arrangement of four words and the positioning of the bits that are removed from some of the data packets.

When a trace source is changed the appropriate flag bit, F, is set as:

1 ID.

0 Data on the following byte.

The second byte is always data and the corresponding bit at the end of the sequence, bits A-J, indicates whether this second byte corresponds to the new ID or the previous ID.

If the trace source has not changed in eight formatter frames, an ID change is indicated, even though the new ID is the current ID value. This is done to ensure that the external TCD log has a record in its buffer of the existing trace source.

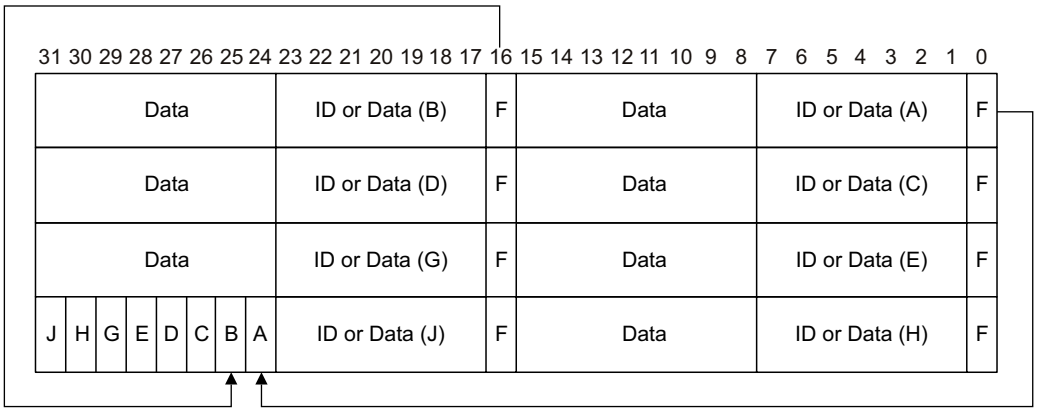


Figure 8-3 Construction of formatter data packets

For more information on the formatter protocol, see the *CoreSight Architecture Specification*.

8.9.2 Special trace source IDs

The following IDs are set aside for special purposes and must not be used under normal operation:

`0x00` NULL trace source. Typically this is not used except when draining the formatter where empty frame locations must be filled.

`0x70-0x7C` Reserved.

`0x7D` Trigger event.

`0x7E` Reserved.

`0x7F` Reserved. This must never be used as a trace source ID because this can cause issues with correctly detecting synchronization packets.

8.9.3 Supported modes of operation

The formatter within the TPIU supports the following basic modes of operation:

Bypass IDs are not incorporated and raw trace data is emitted. It is assumed that the trace source does not change. This requires the use of **tracectl**.

Normal This mode supports multiple trace sources in a single data stream. The **tracectl** pin indicates when data is valid.

Continuous This mode supports multiple trace sources in a single data stream. Data is output continuously, and the information about when data is valid is included in the data stream. Also, triggers are embedded into the data stream because they cannot be indicated otherwise.

For more information on these modes, see the *CoreSight Architecture Specification*.

8.9.4 Periodic synchronization

When the formatter is in continuous mode, it can output more synchronization packets, both intraframe and interframe. When an interframe synchronization packet is emitted in continuous mode, this causes the synchronization counter to restart counting. Full and half synchronization packets are output within interframe and intraframe respectively. For more information, see the *CoreSight Architecture Specification*.

8.10 Configuration options

Existing TPAs that are only capable of operation with **tracectl** must only use the formatter in either bypass or normal mode, not continuous.

8.10.1 Configuration guidelines

The TPIU configuration guidelines are as follows:

- ARM recommends that following a trigger event within a multi-trace source configuration, a flush must be performed to ensure that all historical information related to the trigger is output.
- If Flush on Trigger Event and Stop on Trigger Event options are chosen then any data after the trigger is not captured by the TPA. When the TPIU is instructed to stop, it returns **atreadys** HIGH and does not store any of the accepted data.
- Although multiple flushes can be scheduled using Flush on Trigger Event, Flush on **flushin** and manual flush, when one of these requests are made it masks additional requests of the same type. This means repeated writing to the manual flush bit does not schedule multiple manual requests unless each is permitted to complete first.
- Unless multiple triggers are required, it is not advisable to set both Trigger on Trigger Event and Trigger on Flush Completion, if Flush on Trigger Event is also enabled. In addition, if Trigger on **trigin** is enabled with this configuration, it can also cause multiple trigger markers from one trigger request.

8.11 Example configuration scenarios

This section describes a number of configurations scenarios:

- *Capturing trace after an event and stopping*.
- *Only indicating triggers and still flushing* on page 8-18.
- *Multiple trigger indications* on page 8-18.
- *Independent triggering and flushing* on page 8-18.

8.11.1 Capturing trace after an event and stopping

Two things must happen before trace capture can be stopped:

- A suitable length of time has to elapse to encompass knock-on effects within trace data.
- All historical information relating to these previous events must have been emitted.

Figure 8-4 shows a possible time-line of events where an event of interest, referred to as a trigger event, causes some trace that must be captured and thereafter the trace capture device can be stopped.

When one trace source is used, there is no requirement to flush the system. Instead, the length of the trigger counter delay can be increased to enable more trace to be generated, thereby pushing out historical information.

Traditionally only the initial trigger event is sent to the TPA at time t1, indicated with **tracectl** and a special encoding on **tracedata**. This can still be done but if trace is stopped at this point, there might still be related trace stalled within the ATB system. Trigger signals are now abstracted from ATB through the CTI or CTM infrastructure. To enable all trace information that can relate to an internally generated trigger event to be output, the system must be flushed. Trace capture can then be safely stopped.

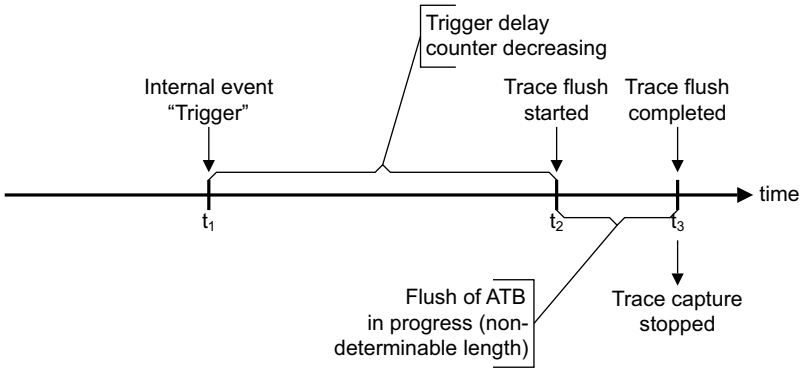


Figure 8-4 Capturing trace after an event and stopping

In Figure 8-4, the action that causes trace capture to be stopped at time t3 can be one of the following:

- The TPA can watch for a trigger to be indicated through **tracectl** and stop.
- The TPA can watch for a trigger to be indicated in the **tracedata** stream, using continuous mode without the requirement for **tracectl**.
- The TPIU can automatically stop trace if the TPA only recognizes trace disable cycles, for example, if it cannot act on trigger markers.

8.11.2 Only indicating triggers and still flushing

It is possible to indicate a trigger at the soonest possible moment and cause a flush while at the same time still permitting externally requested flushes. This enables trace around a key event to be captured and all historical information to be stored within a period immediately following the trigger. Use a secondary event to cause regular trace flushes.

8.11.3 Multiple trigger indications

Sending a trigger to external tools can have additional consequences apart from stopping trace capture. For example, in cases where the events immediately before the trigger might be important but only a small buffer is available, uploads to a host computer for decompression can occur, therefore reducing the amount stored in the TPA. This is also useful where the trigger originated from a device not directly associated with a trace source and is a marker for a repeating interesting event. Figure 8-5 shows multiple trigger indications from flushes.

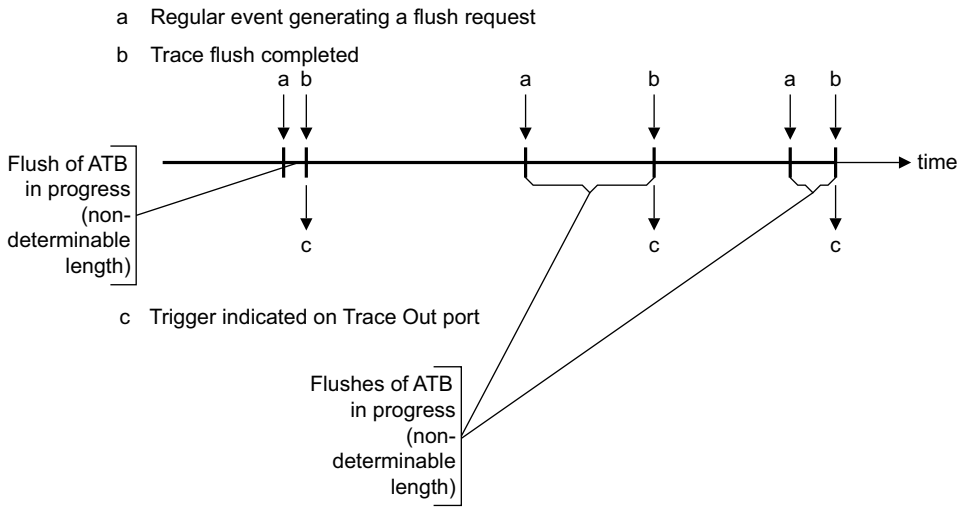


Figure 8-5 Multiple trigger indications from flushes

8.11.4 Independent triggering and flushing

The TPIU has separate inputs for flushes and triggers and, although one can be configured to generate the other, there might be a requirement to keep them separate. To enable a consistent flow of new information through the Trace Out port, there might be a regular flush scheduled, generated from a timing block connected to a CTI. These regular events must not be marked in the trace stream as triggers. Special events coming through the CTI that require a marker must be passed through the **trigin** pin that can either be immediately indicated or, as shown in Figure 8-6 on page 8-19, can be delayed through other flushes and then indicated to the TPA.

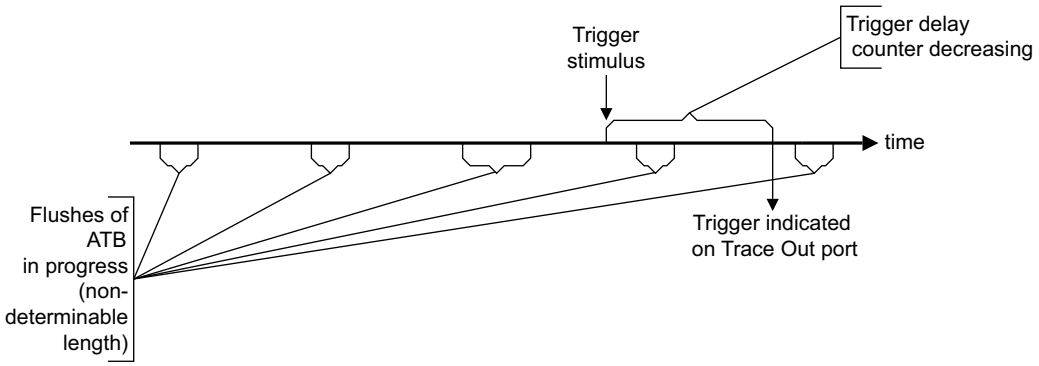


Figure 8-6 Independent triggering during repeated flushes

Chapter 9
Embedded Trace Buffer

This chapter describes the ETB for CoreSight. It contains the following sections:

9.1 About the ETB

The ETB provides on-chip storage of trace data using 32-bit RAM. Figure 9-1 shows the main ETB blocks.

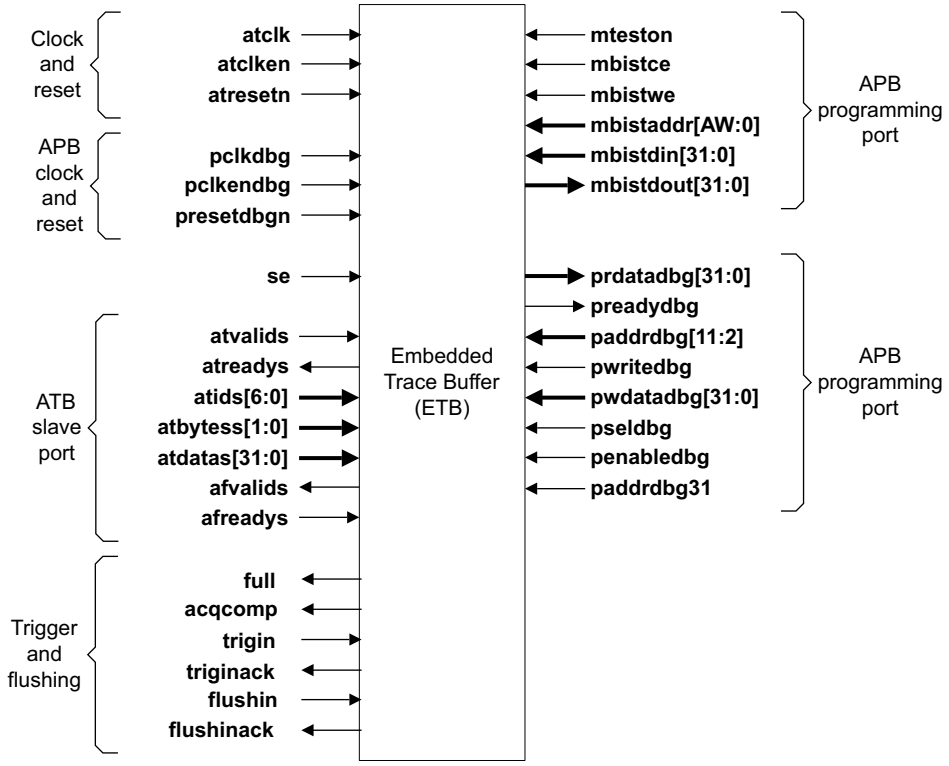


Figure 9-1 ETB block diagram

The ETB accepts trace data from the CoreSight trace source components through an *AMBA Trace Bus* (ATB). For more information, see the *CoreSight Architecture Specification*.

The ETB contains the following blocks:

Formatter Inserts source ID signals into the data packet stream so that trace data can be re-associated with its trace source after the data is read back out of the ETB.

Control Control registers for trace capture and flushing.

APB interface

Read, write, and data pointers provide access to ETB registers. In addition, the APB interface supports wait states through the use of a **preadydbg** signal which is output by the ETB.

The APB interface is synchronous to the ATB domain.

Register bank

Contains the management, control, and status registers for triggers, flushing behavior, and external control.

Trace RAM interface

Controls reads and writes to the trace RAM.

***Memory Built-In Self Test* (MBIST) interface**

Provides test access to the trace RAM.

9.1.1 ATB interface

The ETB accepts trace data from any CoreSight-compliant component trace source with an ATB master port, such as a trace source or a trace funnel.

9.1.2 ETB triggering and flushing ports

Table 9-1 shows the ETB triggering and flushing ports.

Table 9-1 ETB triggering and flushing ports

Name	Type	Description
trigin	Input	From either a CTI or direct from a trace source. Used to enable the trigger to affect the captured trace stream.
triginack	Output	Return response to the CTI. Acknowledgement to **trigin**.
flushin	Input	External control used to assert the ATB signal **afvalids** and drain any historical FIFO information on the bus.
flushinack	Output	Return acknowledgement to **flushin**.

9.1.3 ETB status ports

The ETB continuously captures and writes data into RAM when trace capture is enabled and either:

- The trigger counter is static at 0.
- The trigger counter has not yet decremented to zero. The trigger counter begins to decrement when a trigger is observed.

A trigger event occurs when the trigger counter reaches zero, or if the trigger counter is zero when the trigger input is HIGH.

When the trigger counter reaches zero the acquisition complete flag, `AcqComp`, is activated and trace capture stops. The value loaded into the trigger counter therefore sets the number of data words stored in the trace RAM after a trigger event.

Table 9-2 shows the ETB status ports.

Table 9-2 ETB status ports

Name	Type	Description
acqcomp	Output	When HIGH, indicates that trace acquisition is complete, and the trigger counter is at zero.
full	Output	When HIGH, indicates that the ETB RAM has overflowed or wrapped around to address zero.

9.1.4 Memory BIST interface

Table 9-3 on page 9-4 shows the Memory BIST interface ports.

Table 9-3 ETB Memory BIST interface ports

Name	Type	Description
mbistaddr [CSETB_ADDR_WIDTH-1:0]	Input	Address bus for the external BIST controller, active when **mteston** is HIGH. **CSETB_ADDR_WIDTH** defines the address bus width used, and therefore the RAM depth supported.
mbistce	Input	Active-HIGH chip select for external BIST controller, active when **mteston** is HIGH.
mbistdin[31:0]	Input	Write data bus for external BIST controller, active when **mteston** is HIGH.
mbistdout[31:0]	Output	Read data bus for external BIST controller, active when **mteston** is HIGH.
mbistwe	Input	Active-HIGH write enable for external BIST controller, active when **mteston** is HIGH.
mteston	Input	Enable signal for the external BIST controller.

9.2 ETB clocks, resets, and synchronization

This section describes ETB clocks, resets, and synchronization in:

- *ETB clock domains*.
- *ETB resets*.
- *ETB synchronization*.

9.2.1 ETB clock domains

The ETB has the following clock domains:

pclkdbg Drives the APB interface and selected control registers.

atclk Drives the ATB interface, all control logic, and RAM.

9.2.2 ETB resets

atresetn resets all of the ETB registers in the **atclk** domain. **atresetn** must be synchronized externally in the CoreSight infrastructure by a global CoreSight reset signal that asserts **atresetn** asynchronously and deasserts **atresetn** synchronously with **atclk**.

presetdbgn resets the APB interface of the ETB. **presetdbgn** must be synchronized externally in the CoreSight infrastructure by a global CoreSight reset signal that asserts **presetdbgn** asynchronously and deasserts **presetdbgn** synchronously with **pclkdbg**.

9.2.3 ETB synchronization

atclk and **pclkdbg** are synchronous domains.

Note

pclkdbg must be equivalent to, or an integer division of, **atclk**.

9.3 ETB trace capture and formatting

The formatter inserts the source ID signal **atids[6:0]** into a special format data packet stream to enable trace data to be reassociated with a trace source after data is read back out of the ETB.

9.3.1 Formatter data processing

Figure 9-2 shows the arrangement of four words and the organization of the data packets.

When a trace source is changed the appropriate flag bit, F, is set to:

1 ID.

0 Data on the first byte.

The second byte is always data. The corresponding bit at the end of the sequence, bits A to J, indicates whether this second byte corresponds to the new ID, that is, bit clear, or the previous ID, that is, bit set.

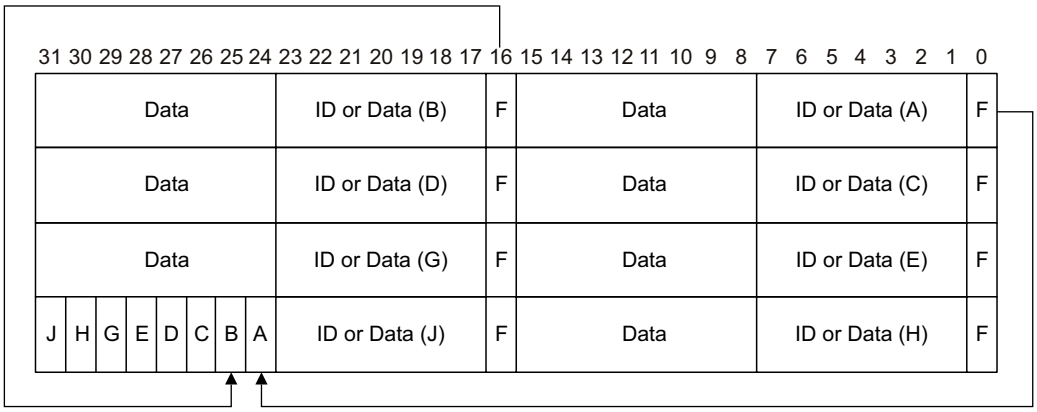


Figure 9-2 Construction of data packets within the formatter

9.3.2 Special trace source IDs

The following IDs are set aside for special purposes and must not be used under normal operation:

`0x00` This indicates a NULL trace source. It identifies unused data packets within the formatter so that incomplete packets can be stored into RAM when data capture ceases.

`0x70-0x7C` Reserved.

`0x7D` Allocated for indication of a trigger within the trace stream.

`0x7E` Reserved.

`0x7F` Reserved. This ID must never be used as a trace source ID because it can cause issues with correctly detecting the synchronization packet.

9.3.3 Special modes of operation

The formatter supports the following distinct modes of operation as specified by bits[1:0] in the FFCR:

Bypass In this mode, no formatting information is inserted into the trace stream and a raw reproduction of the incoming trace stream is stored. If any bytes remain in the formatter when tracing is stopped, because of non 32-bit **atdatas**, then the word stored to RAM is appended with a single logic 1 bit and filled in with zeros. A second word, `0x00000000` is then stored. If no bytes remain in the formatter, when capture is stopped, four additional words are stored, `0x00000001` followed by three words of `0x00000000`.

When data is later decompressed, it is possible to determine that a post-amble is present by back-tracking the trailing zero data at then end of the trace stream until the last single bit at logic 1 is detected. All data preceding this first logic 1 is then treated as decompressible data. When all data has been stored in the RAM, FtStopped in the FFSR is set HIGH.

Note

This mode assumes that the source ID does not change.

Normal Normal mode has the option of embedding triggers into the data packets.

Formatting information is added to indicate the change of source ID along with the associated wrapping additions, as described in *TPIU formatter and FIFO* on page 8-14. If any data remains in the formatter when tracing is stopped, the formatter is filled with NULL packets, ID = `0x00`, to ensure that all data is stored in RAM. When all remaining data has been stored in the RAM, **FtStopped** in the FFSR is set HIGH.

Continuous Continuous mode in the ETB corresponds to normal mode with the embedding of triggers. Unlike continuous mode in the TPIU, no formatter synchronization packets are added because alignment of data to the RAM locations is assumed.

9.3.4 Stopping trace

The stopping of trace is indicated by **FtStopped** set to HIGH in the FFSR. This is set by:

- **TraceCaptEn** = 0 in the Control Register.
- A trigger event occurring, **trigin** goes HIGH and the Trigger Counter Register reaches zero, and **StopTrig** is set to HIGH in the FFCR, `0x304`.
- A flush completing and **StopFl** is set to HIGH in the FFCR.

StopTrig and **StopFl** in the FFCR can be enabled at the same time. For example, if **FOnTrig** in the FFCR is set to perform a flush on a trigger event, but **StopTrig** is HIGH, none of the flushed data is stored, because **StopTrig** is HIGH. If the situation requires that all the flushed data is captured, then **StopTrig** is LOW and **StopFl** is HIGH.

When the formatter stops, **atreadys** is output HIGH to prevent an ATB bus from stalling. This is important when a replicator is present, but the received data is ignored.

9.4 Flush assertion

All three flush generating conditions can be enabled together:

- Flush from **flushin**.
- Flush from trigger.
- Manually activated flush.

If more flush events are generated while a flush is in progress, the current flush is serviced before the next flush is started. Only one request for each source of flush can be pended. If a subsequent flush request signal is deasserted while the flush is still being serviced or pended, a second flush is not generated.

Flush from **flushin** takes priority over flush from trigger, which in turn is completed before a manually activated flush. For more information, see *Additional reading* on page ix.

9.5 Triggers

A trigger event is defined as when:

- The trigger counter reaches zero.
- The trigger counter is zero and **trigin** is HIGH.

All trigger indication conditions, TrigEvt, TrigFl, and TrigIn, in the FFCR can be enabled simultaneously. This results in multiple triggers appearing in the trace stream.

The trigger counter register controls how many words are written into the trace RAM after a trigger event. After the formatter is flushed in normal or continuous mode, a complete empty frame is generated. This is a data overhead of seven extra words in the worst case. The trigger counter defines the number of 32-bit words remaining to be stored in the ETB trace RAM. If the formatter is in bypass mode, a maximum of two additional words are stored for the trace capture post-amble. See Chapter 3 *Programmers Model*.

9.6 Write address generation for trace data storage

The RAM Write Pointer is automatically selected to address the trace RAM when trace capture is enabled in the Control Register. Data is written into the trace RAM when the formatter asserts the data valid flag on generation of a formatted word. On completion of a write access to the trace RAM, the register is automatically incremented.

The RAM Write Pointer Register must be programmed before trace capture is enabled.

9.7 Trace data storage

The data is received by the ETB ATB formatter block. When **TraceCaptEn** is asserted, the formatter starts the process of embedding the source IDs into the data stream, or normal mode, and then outputs 32-bit words to be stored in the ETB trace RAM. When data is ready to be stored in the trace RAM, data is written to the address stored in the RAM Write Pointer Register. When the Trigger Counter Register reaches zero, the acquisition complete flag **AcqComp** is asserted and trace capture is stopped.

9.8 APB configuration and RAM access

This section describes APB configuration and RAM accesses:

- *Read access*.
- *Write access*.

9.8.1 Read access

When trace capture is disabled, TraceCaptEn=0, the RAM Read Pointer is used to generate the RAM address for reading data stored in the ETB trace RAM. Updating the RAM Read Pointer automatically triggers a RAM access to ensure that all RAM data present in the RAM Read Data Register is up-to-date.

The RAM Read Pointer is updated either by writing directly to it, or performing a read of the RAM Read Data Register, causing the RAM Read Pointer to be incremented. Therefore, a read of the RAM Read Data Register causes the next memory location to be read and loaded into the RAM Read Data Register.

9.8.2 Write access

When trace capture is disabled, TraceCaptEn=0, the RAM Write Pointer Register is used to generate the RAM address for writing data from the RAM Read Data Register. The data to be stored in the RAM Write Data Register is derived from an APB write transfer to the APB interface.

Updating the RAM Read Data Register automatically triggers a RAM access to write the register contents into the trace RAM at the address present in the RAM Write Pointer Register. On completion of the write cycle the RAM Write Pointer Register is automatically incremented.

Note

A write access to the RAM Write Pointer Register does not initiate a write transfer to the trace RAM. Only a write to the RAM Read Data Register causes a RAM write.

Program the RAM Write Pointer Register before trace capture is enabled again.

9.9 Trace RAM

See *Additional reading* on page ix for information about trace RAM and RAM integration.

9.10 Authentication requirements for CoreSight ETBs

The CoreSight ETBs do not require any inputs capable of disabling them.

9.11 ETB RAM support

The following sections describe the ETB RAM support:

- *Access sizes*.
- *BIST interface*.
- *RAM instantiation*.

9.11.1 Access sizes

Byte writable RAMs are not supported. Trace storage is only in word accesses, and APB accesses take place through the 32-bit RAM Write Data Register. The APB interface has no concept of data size.

9.11.2 BIST interface

The RAM BIST interface connects though the trace RAM Interface block to provide test access to the trace RAM. **mteston** enables the memory BIST interface and disables all other accesses to and from the RAM.

9.11.3 RAM instantiation

As shown in Figure 9-3, the cxetbram block provides the wrapper in which the RAM simulation model or hardened cell can be instantiated. The active level of chip enable, write enable can be modified in this block.

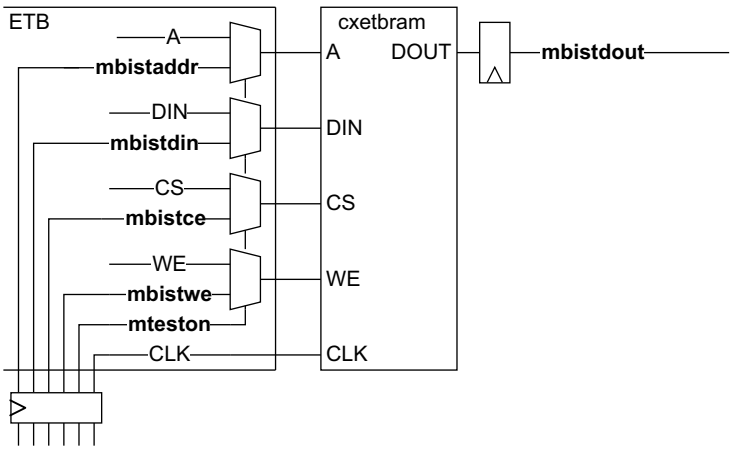


Figure 9-3 ETB trace RAM block wrapper

Chapter 10
Granular Power Requestor

This chapter describes the Granular power requestor and contains the following section:

- *Granular Power Requestor interfaces* on page 10-2.

10.1 Granular Power Requestor interfaces

This section has the following sub-sections:

- *Clock and reset*.
- *Functional interfaces*.
- *Device unlocking*.

10.1.1 Clock and reset

This component has a single clock domain, **clk** that the debug APB clock drives, and a single asynchronous active-LOW reset input, **resetn**.

10.1.2 Functional interfaces

This component has an APB3-compliant slave interface and **CPWRUP** master interfaces.

CPWRUP interface

The **CPWRUP** interface is an asynchronous request and acknowledge interface that enables a requester to communicate with a power controller through a 4-phase handshake mechanism.

Memory-mapped registers in GPR control the CPWRUP interface ports.The GPR has hardware logic that enforces the appropriate protocol for the 4-phase handshake.

GPR programming interface

The GPR has 4KB memory map footprint and contains CoreSight management registers. See *Additional reading* on page ix. The DEVTYPE Register of GPR returns `0x34` on a read operation to indicate that it is a power requestor block. This value is derived from the register fields:

- MAJOR = `0x4`, meaning Debug Control.
- SUB = `0x3`, meaning Debug Power Requestor.

The APB interface supports zero-wait-state write operations and single-wait-state read operations on the APB.

10.1.3 Device unlocking

On reset, the device is locked. The device is unlocked when you write `32'hC5ACCE55` to the LAR. Write operations to other registers in the device are permitted only when the device is unlocked. Read operations are permitted regardless of the device lock status. When **paddr31** is driven HIGH and the debugger initiates an operation:

- Write operations to the LAR are ignored.
- Read operations to the LAR return 0, indicating that no lock mechanism is present.

Appendix A
Signal Descriptions

This appendix describes the CoreSight SoC port and interface signals. It contains the following sections:

A.1 Debug Access Port signals

The section describes the DAP signals.It has the following subsections:

A.1.1 Serial wire or JTAG - Debug Port signals

Table A-7 on page A-10 shows the *Serial Wire or JTAG Debug Port* (SWJ-DP) signals.

Table A-1 Serial wire and JTAG debug port signals

Name	Type	Clock domain	Description
swdo	Output	swclktck	Serial wire data output.
swdoen	Output	swclktck	Serial wire data output enable.
tdo	Output	swclktck	JTAG TAP data ut.
ntdoen	Output	swclktck	TAP data out enable.
dapcaddr[15:2]	Output	dapclk	Compressed DAP address.
dapwrite	Output	dapclk	DAP bus write.
dapenable	Output	dapclk	DAP enable transaction.
dapabort	Output	dapclk	DAP abort.
dapsel	Output	dapclk	DAP transaction select.
dapwdata[31:0]	Output	dapclk	DAP write data.
cdbgpwrupreq	Output	NA	Debug power domain powerup request.
csyspwrupreq	Output	NA	System power domain powerup request.
cdbgrstreq	Output	NA	Debug reset request to reset controller.
jtagnsw	Output	swclktck	**HIGH** If JTAG selected. **LOW** If SWD selected.
jtagtop	Output	swclktck	JTAG state machine is in one of the four modes: • Test-Logic-Reset. • Run-Test/Idle. • Select-DR-Scan. • Select-IR-Scan.
ntrst	Input	swclktck	TAP asynchronous reset.
npotrst	Input	swclktck	JTAG power-on reset and Serial wire power-on reset.
swclktck	Input	NA	Serial wire clock and TAP clock.
swditms	Input	swclktck	Serial wire data input and TAP test mode select.
tdi	Input	swclktck	JTAG TAP data in or alternative input function.
dapresetn	Input	dapclk	DAP asynchronous reset active-LOW.
dapclk	Input	dapclk	DAP clock.
dapclken	Input	dapclk	DAP clock enable.
daprdata[31:0]	Input	dapclk	DAP read data.
dapready	Input	dapclk	DAP data bus ready.
dapslverr	Input	dapclk	AP slave error response.
cdbgpwrupack	Input	None	Debug power domain up acknowledge.

Table A-1 Serial wire and JTAG debug port signals (continued)

Name	Type	Clock domain	Description
csyspwrupack	Input	None	System power domain up acknowledge.
cdbgrstack	Input	None	Debug reset acknowledge to reset controller.
targetid[31:0]	Input	None	Target ID for SW multi-drop selection.
instanceid[3:0]	Input	None	Instance ID for SW multi-drop selection.

A.1.2 DAP bus interconnect signals

Table A-7 on page A-10 shows the DAP bus interconnect signals.

Table A-2 DAP bus interconnect signals

Name	Type	Clock domain	Description
clk	Input	clk	DAP Clock.
resetn	Input	clk	DAP Reset.
dapcaddrm<x>[7:2]	Output	clk	DAP compressed address bus.
dapselm<x>	Output	clk	DAP select.
dapenablem<x>	Output	clk	DAP enable.
dapwritem<x>	Output	clk	DAP write or read.
dapwdatam<x>[31:0]	Output	clk	DAP write data bus.
dapabortm<x>	Output	clk	DAP abort.
daprdatam31<x>[31:0]	Input	clk	DAP read data bus.
dapreadym<x>	Input	clk	DAP ready.
dapslverrm<x>	Input	clk	DAP error.
dapcaddrs[15:2]	Input	clk	DAP compressed address bus.
dapsels	Input	clk	DAP select.
dapenables	Input	clk	DAP enable.
dapwrites	Input	clk	DAP write or read.
dapwdatas[31:0]	Input	clk	DAP write data bus.
dapaborts	Input	clk	DAP abort.
daprdatas[31:0]	Output	clk	DAP read data bus.
dapreadys	Output	clk	DAP ready.
dapslverrs	Output	clk	DAP error.

A.1.3 DAP asynchronous bridge signals

Table A-3 shows the DAP asynchronous bridge signals.

Table A-3 DAP asynchronous bridge signals

Signal	Type	Clock domain	Description
dapclks	Input	dapclks	DAP clock.
dapclkens	Input	dapclks	DAP clock enable.
dapresetsn	Input	dapclks	DAP reset.
dapclkenm	Input	dapclks	DAP clock enable.
dapclkm	Input	dapclkm	DAP clock.

Table A-3 DAP asynchronous bridge signals (continued)

Signal	Type	Clock domain	Description
dapresetm	Input	dapclks	DAP reset.
dapsels	Input	dapclks	The DAP select signal. Indicates that the DAP bus master is selecting this slave device and requires a data transfer.
dapaborts	Input	dapclks	The DAP abort. When the bus master asserts **dapaborts** HIGH, the DAP slave aborts the present DAP transfer and asserts **dapreadys** HIGH in the next cycle.
dapenables	Input	dapclks	The DAP enable. Indicates the second and subsequent cycles of a DAP transfer.
dapwrites	Input	dapclks	The DAP RW select signal. It indicates a DAP write access when HIGH and a DAP read access when LOW.
dapaddrs[31:0]	Input	dapclks	The DAP address bus.
dapwdatas[31:0]	Input	dapclks	The DAP write data bus.
dapreadys	Output	dapclks	The DAP ready. The DAP bus slave asserts this signal HIGH to indicate that it completed the current DAP transfer and is ready for the next transfer.
dapslverrs	Output	dapclks	The DAP slave error. When HIGH, the DAP slave indicates that the present DAP transaction had an error.
daprdatas[31:0]	Output	dapclks	The DAP read data. Carries the read-data of a DAP read transfer.
dapselm	Output	dapclkm	The DAP select. Indicates that the master is selecting a particular slave for RW transfer.
dapabortm	Output	dapclkm	The DAP abort. When asserted HIGH, it indicates that the master is aborting the present transaction.
dapenablem	Output	dapclkm	The DAP enable. Indicates the second and subsequent cycles of a DAP transfer.
dapwritem	Output	dapclkm	The DAP RW. Indicates a write transfer when HIGH and a read transfer when LOW.
dapaddrm[31:0]	Output	dapclkm	The DAP address bus.
dapwdatam[31:0]	Output	dapclkm	The DAP write data. The master drives this bus and carries the write data for the present write transfer.
dapreadym	Input	dapclkm	The DAP ready. The slave indicates whether it has completed present transfer and is ready for the next transfer.
dapslverrm	Input	dapclkm	The DAP slave error. The slave indicates that the present transfer has in error.
daprdatam[31:0]	Input	dapclkm	The DAP read data. The read data of the present DAP read transfer.
csysreq[a]	Input	dapclks	Clock powerdown request.
csysack[a]	Output	dapclks	Clock powerdown acknowledge.
cactive[a]	Output	dapclks	Clock is required when driven HIGH.

a. This signal is only present if you configure this component to have an LPI.

A.1.4 DAP synchronous bridge signals

Table A-4 on page A-7 shows the DAP synchronous bridge signals.

Table A-4 DAP synchronous bridge signals

Signal	Type	Clock domain	Description
dapclk	Input	dapclk	The DAP clock.
dapresetn	Input	dapclk	The DAP reset.
dapclkens	Input	dapclk	The DAP clock enable.
dapsels	Input	dapclk	The DAP select. Indicates that the slave device is selected and a data transfer is required.
dapaborts	Input	dapclk	The DAP abort. When this signal is asserted HIGH, then the DAP slave aborts the current DAP transfer and assert its ready in the next cycle.
dapenables	Input	dapclk	The DAP enable. Indicates the second and subsequent cycles of an DAP transfer
dapwrites	Input	dapclk	The DAP RW. Indicates a DAP write access when HIGH and a DAP read access when LOW.
dapaddrs[31:0]	Input	dapclk	The DAP address bus from master.
dapwdatas[31:0]	Input	dapclk	The DAP write data. The DAP master drives this bus.
dapreadys	Output	dapclk	The DAP slave ready. When asserted HIGH, it indicates that the slave completed the present DAP transfer and is ready for the next transfer.
dapslverrs	Output	dapclk	The DAP slave error. Indicates that the present DAP transaction has an error.
daprdatas[31:0]	Output	dapclk	The DAP read data. Carries the read data of a DAP read transfer.
dapclkenm	Input	dapclk	The DAP clock enable.
dapselm	Output	dapclk	The DAP select. Indicates that the master is selecting a particular slave for RW transfer.
dapabortm	Output	dapclk	The DAP master abort. When asserted HIGH, it indicates that the master is aborting the current transaction.
dapenablem	Output	dapclk	The DAP enable. Indicates the second and subsequent cycles of a DAP transfer.
dapwritem	Output	dapclk	The DAP RW. Indicates a write transfer when HIGH and a read transfer when LOW.
dapaddrm[31:0]	Output	dapclk	The DAP address bus.
dapwdatam[31:0]	Output	dapclk	The DAP write data. The master drives this bus and carries the write data for the current write transfer.
dapreadym	Input	dapclk	The DAP ready. The slave indicates whether it has completed current transfer and is ready for the next transfer.
dapslverrm	Input	dapclk	The DAP slave error. The slave indicates that the present transfer has an error.
daprdatam[31:0]	Input	dapclk	The DAP read data. The read data of the present DAP read transfer.
csysreq	Input	dapclk	Clock powerdown request.
csysack	Output	dapclk	Clock powerdown acknowledge.
cactive	Output	dapclk	Clock is required when driven HIGH.

A.1.5 JTAG - Access Port signals

Table A-7 on page A-10 shows the JTAG - Access Port signals.

Table A-5 DAP JTAG access port signals

Name	Type	Clock domain	Description
dapclk	Input	dapclk	DAP internal clock.
dapclken	Input	dapclk	DAP clock enable.
dapresetn	Input	dapclk	DAP reset.
dapsel	Input	dapclk	DAP select.
dapenable	Input	dapclk	DAP enable.
dapwrite	Input	dapclk	DAP read or write.
dapabort	Input	dapclk	DAP abort.
dapcaddr[7:2]	Input	dapclk	DAP compressed address bus.
dapwdata[31:0]	Input	dapclk	DAP write data.
csrtck[7:0]	Input	dapclk	Returns TCK from JTAG slaves.
srstconnected[7:0]	Input	dapclk	JTAG slaves support of SRST.
portconnected[7:0]	Input	dapclk	JTAG slaves support of SRST.
portenabled[7:0]	Input	dapclk	JTAG slaves support of SRST.
cstdo[7:0]	Input	dapclk	TDO from JTAG slaves.
dapready	Output	dapclk	DAP ready.
daprdata[31:0]	Output	dapclk	DAP read data.
nsrstout[7:0]	Output	dapclk	Subsystem reset to JTAG slaves.
ncstrst[7:0]	Output	dapclk	Test reset to JTAG slaves.
cstck[7:0]	Output	dapclk	Test clock to JTAG slaves.
cstdi[7:0]	Output	dapclk	Test data input to JTAG slaves.
cstms[7:0]	Output	dapclk	Test mode select to JTAG slaves.

A.1.6 Advanced eXtensible Interface - Access Port signals

Table A-7 on page A-10 shows the DAP AXI access port signals.

Table A-6 DAP AXI access port signals

Name	Type	Clock domain	Description
clk	Input	clk	Clock.
resetn	Input	clk	Reset.
dapaddr[7:2]	Input	clk	DAP address bus.
dapsel	Input	clk	DAP select. Asserted when the master selects this DAP slave and requires a data transfer.

Table A-6 DAP AXI access port signals (continued)

Name	Type	Clock domain	Description
dapenable	Input	clk	The DAP enable. Indicates the second and subsequent cycles of a DAP transfer.
dapwrite	Input	clk	The DAP RW. Indicates a DAP write access when HIGH and a DAP read access when LOW.
dapwdata[31:0]	Input	clk	The DAP write data.
dapabort	Input	clk	The DAP abort. When this signal is asserted HIGH, then the DAP slave aborts the present DAP transfer and asserts its ready in the next cycle.
daprdata[31:0]	Output	clk	The DAP read data. Carries the read data of a DAP read transfer.
dapready	Output	clk	The DAP ready. When asserted HIGH, it indicates that the slave completed the present DAP transfer and is ready for the next transfer.
dapslverr	Output	clk	The DAP slave error. Indicates that the present DAP transaction has an error.
dbgen	Input	clk	Debug enable for AHB transfers.
spiden	Input	clk	Secure Privileged Invasive Debug Enable. Prevents secure transfer initiation when LOW.
awaddr[63:0]	Output	clk	Write address.
awlen[3:0]	Output	clk	Write burst length.
awsize[2:0]	Output	clk	Write burst size.
awburst[1:0]	Output	clk	Write burst type.
awlock	Output	clk	Write lock type.
awcache[3:0]	Output	clk	Write cache type.
awprot[2:0]	Output	clk	Write protection type.
awvalid	Output	clk	Write address valid.
awready	Input	clk	Write address ready.
wdata[63:0]	Output	clk	Write data.
wstrb[7:0]	Output	clk	Write byte-lane strobes.
wlast	Output	clk	Write data last transfer indication.
wvalid	Output	clk	Write data valid.
wready	Input	clk	Write data ready.
bresp[1:0]	Input	clk	Write response.
bvalid	Input	clk	Write response valid.
bready	Output	clk	Write response ready.
araddr[63:0]	Output	clk	Read address.
arlen[3:0]	Output	clk	Read burst length.
arsize[2:0]	Output	clk	Read burst size.
arburst[1:0]	Output	clk	Read burst type.

Table A-6 DAP AXI access port signals (continued)

Name	Type	Clock domain	Description
arlock	Output	clk	Read lock type.
arcache[3:0]	Output	clk	Read cache type.
arprot[2:0]	Output	clk	Read protection type.
arvalid	Output	clk	Read address valid.
arready	Input	clk	Read address ready.
rdata[63:0]	Input	clk	Read data.
rresp[1:0]	Input	clk	Read data response.
rlast	Input	clk	Read data last transfer indication.
rvalid	Input	clk	Read data valid.
rready	Output	clk	Read data ready.
awdomain[1:0]	Output	clk	Write domain.
awsnoop[2:0]	Output	clk	Write snoop request type.
awbar[1:0]	Output	clk	Write barrier type.
ardomain[1:0]	Output	clk	Write domain.
arsnoop[3:0]	Output	clk	Read snoop request type.
arbar[1:0]	Output	clk	Read barriers.
rombaseaddrl[31:0]	Input	clk	Most significant 32-bit of the AXI-AP Debug Base Address Register.
rombaseaddru[31:0]	Input	clk	Least significant 32-bit of the AXI-AP Debug Base Address Register.

A.1.7 Advanced High-performance Bus - Access Port signals

Table A-7 shows the DAP AHB access port signals.

Table A-7 DAP AHB access port signals

Signal	Type	Clock domain	Description
dapabort	Input	dapclk	DAP abort.
dapcaddr[7:2]	Input	dapclk	DAP compressed address bus.
dapclk	Input	dapclk	DAP clock.
dapclken	Input	dapclk	DAP clock enable.
dapenable	Input	dapclk	DAP enable.
dapresetn	Input	dapclk	DAP reset.
dapsel	Input	dapclk	DAP select.
dapwdata[31:0]	Input	dapclk	DAP write data bus.
dapwrite	Input	dapclk	DAP write or read.
dbgen	Input	dapclk	Debug enable for AHB transfers.

Table A-7 DAP AHB access port signals (continued)

Signal	Type	Clock domain	Description
hrdatam	Input	dapclk	AHB read data bus.
hreadym	Input	dapclk	AHB slave ready.
hrespm	Input	dapclk	AHB slave response.
spiden	Input	dapclk	Secure Privileged Invasive Debug Enable. Prevents secure transfer initiation when LOW.
rombaseaddr[31:0]	Input		Static port to define the AHB ROM table base address.
daprdata[31:0]	Output	dapclk	DAP read data bus.
dapready	Output	dapclk	DAP ready.
dapslverr	Output	dapclk	DAP slave error.
haddrm[31:0]	Output	dapclk	AHB address bus.
hbstrbm[3:0]	Output	dapclk	AHB byte lane strobe.
hburstm[2:0]	Output	dapclk	AHB burst type.
hlockm	Output	dapclk	AHB lock transfer.
hprotm[6:0]	Output	dapclk	AHB protection type.
hsizem[2:0]	Output	dapclk	AHB transfer size.
htransm[1:0]	Output	dapclk	AHB transfer type.
hwdatam[31:0]	Output	dapclk	AHB write data bus.
hwritem	Output	dapclk	AHB write or read.

A.1.8 Advanced Peripheral Bus - Access Port signals

Table A-8 shows the APB access port signals.

Table A-8 APB access port signals

Name	Type	Clock domain	Description
dapclk	Input	dapclk	DAP clock.
dapclken	Input	dapclk	DAP clock enable.
dapresetn	Input	dapclk	DAP reset.
dapsel	Input	dapclk	DAP select.
dapenable	Input	dapclk	DAP enable.
dapwrite	Input	dapclk	DAP write or read.
dapabort	Input	dapclk	DAP abort.
dapcaddr[7:2]	Input	dapclk	DAP compressed address bus.
dapwdata[31:0]	Input	dapclk	DAP write data bus.
pready	Input	dapclk	APB ready.
pslverr	Input	dapclk	APB slave error.
prdata[31:0]	Input	dapclk	APB read data bus.
deviceen	Input	dapclk	Device enable.
dapready	Output	dapclk	DAP ready.
dapslverr	Output	dapclk	DAP slave error.
daprdata[31:0]	Output	dapclk	DAP read data bus.
psel	Output	dapclk	APB select.
penable	Output	dapclk	APB enable.
pwrite	Output	dapclk	APB write or read.
paddr[31:2]	Output	dapclk	APB address bus.
pwdata[31:0]	Output	dapclk	APB write data bus.
pdbgswen	Output	dapclk	Enable software access to Debug APB.

A.1.9 APB interconnect signals

Table A-9 shows the APB interconnect signals.

Table A-9 APB interconnect signals

Signal	Type	Clock domain	Description
clk	Input	clk	The clock reference signal for all APB debug interfaces. The rising edge of **clk** times all transfers on the APB.
resetn	Input	clk	Active-LOW reset.
paddrm<x>[maw:2][ab]	Output	clk	The APB address bus for master interface <*x*>.

Table A-9 APB interconnect signals (continued)

Signal	Type	Clock domain	Description
pselm<x>[a]	Output	clk	APB select. Indicates that the slave device connected to master interface <*x*> is selected, and a data transfer is required.
penablem<x>[a]	Output	clk	APB enable. Indicates the second and subsequent cycles of an APB transfer that the master interface <*x*> initiates.
pwritem<x>[a]	Output	clk	APB RW transfer. Indicates an APB write access when HIGH, and an APB read access when LOW.
prdatam<x>[31:0][a]	Input	clk	APB read data. Drives this bus during read cycles.
pwdatam<x>[31:0][a]	Output	clk	Write data that the APBIC master interface <*x*> drives.
preadym<x>[a]	Input	clk	APB ready. Uses this signal to extend an APB transfer.
pslverrm<x>[a]	Input	clk	Indicates a transfer failure. The APB peripherals are not required to support the **pslverr** pin.
paddrs<x>[saw:2][cd]	Input	clk	The APB address bus for slave interface <*x*>.
psels<x>[c]	Input	clk	Select. Indicates that the slave interface <*x*> is selected, and a data transfer is required.
penables<x>[c]	Input	clk	Enable. Indicates the second and subsequent cycles of an APB transfer initiated on slave interface <*x*>.
pwrites<x>[c]	Input	clk	Direction. Indicates an APB write access when HIGH and an APB read access when LOW.
prdatas<x>[31:0][c]	Output	clk	Read data. The slave interface <*x*> drives this bus during read cycles.
pwdatas<x>[31:0][c]	Input	clk	Write data that the APB master device connected to the APBIC slave interface <*x*> drives.
preadys<x>[c]	Output	clk	APB ready. The slave interface <*x*> uses this signal to extend an APB transfer.
pslverrs<x>[c]	Output	clk	Indicates a transfer failure.
targetid[31:0]	Input	clk	Provides information about the target when the host is connected to a single device.
paddr31s0	Input	clk	Enables components to distinguish between internal accesses from system software, and external accesses from a debugger.
dbgswen	Input	clk	Enable software access to debug APB.

a. Where <x>=0 to (NUM_MASTER_INTF-1).
b. Where maw is a parameter dependent number.
c. Where <x>=0 to (NUM_SLAVE_INTF-1).
d. Where saw is a parameter dependent number.

A.1.10 APB asynchronous bridge signals

Table A-10 shows the APB asynchronous bridge signals.

Table A-10 APB asynchronous bridge signals

Signal	Type	Clock domain	Description
pclks	Input	pclks	APB clock.
presetsn	Input	pclks	APB reset.
pclkens	Input	pclks	APB clock enable.
pclkm	Input	pclkm	APB clock.
presetmn	Input	pclkm	APB reset.
pclkenm	Input	pclkm	APB clock enable.
psels	Input	pclks	APB select. Indicates that the slave interface is selected and a data transfer is required.
penables	Input	pclks	APB enable. Indicates the second and subsequent cycles of an APB transfer initiated on slave interface.
pwrites	Input	pclks	APB RW transfer. Indicates an APB write access when HIGH and an APB read access when LOW.
paddrs[31:0]	Input	pclks	APB address bus.
pwdatas[31:0]	Input	pclks	APB write data.
preadys	Output	pclks	APB ready. The slave interface uses this signal to extend an APB transfer.
pslverrs	Output	pclks	APB transfer error. Indicates a transfer failure. The APB peripherals are not required to support the **pslverr** pin.
prdatas[31:0]	Output	pclks	APB read data. The slave interface drives this bus during read cycles.
pselm	Output	pclkm	APB select. Indicates that the slave device connected to the master interface is selected and a data transfer is required.
penablem	Output	pclkm	APB enable. Indicates the second and subsequent cycles of an APB transfer that the master interface initiates.
pwritem	Output	pclkm	APB RW transfer. Indicates an APB write access when HIGH and an APB read access when LOW.
paddrm[31:0]	Output	pclkm	APB address bus.
pwdatam[31:0]	Output	pclkm	APB write data. The APB master interface drives this bus.
preadym	Input	pclkm	APB ready. The slave device uses this signal to extend an APB transfer.
pslverrm	Input	pclkm	APB transfer error. Indicates a transfer failure. The APB peripherals are not required to support the **pslverr** pin.
prdatam[31:0]	Input	pclkm	APB read data. The selected slave drives this bus during read cycles.
csysreq[a]	Input	pclks	Clock powerdown request.
csysack[a]	Output	pclks	Clock powerdown acknowledge.
cactive[a]	Output	pclks	Clock is required when driven HIGH.

a. This signal is present only if you configure this component to have an interface of type low-power and slave, or of type low-power and full.

A.1.11 APB synchronous bridge signals

Table A-11 shows the APB synchronous bridge signals.

Table A-11 APB synchronous bridge signals

Signal	Type	Clock domain	Description
pclk	Input	pclk	APB clock signal for all downstream APB debug interfaces.
presetn	Input	pclk	APB reset.
pclkens	Input	pclk	APB clock enable.
psels	Input	pclk	APB select. Indicates that the slave interface is selected and a data transfer is required.
penables	Input	pclk	APB enable. Indicates the second and subsequent cycles of an APB transfer initiated on slave interface.
pwrites	Input	pclk	APB RW transfer. Indicates an APB write access when HIGH and an APB read access when LOW.
paddrs[31:0]	Input	pclk	APB address bus.
pwdatas[31:0]	Input	pclk	APB write data.
preadys	Output	pclk	APB ready. The slave interface uses this signal to extend an APB transfer.
pslverrs	Output	pclk	APB transfer error. Indicates a transfer failure. The APB peripherals are not required to support the **pslverr** pin.
prdatas[31:0]	Output	pclk	APB read data. The slave interface drives this bus during read cycles.
pclkenm	Input	pclk	APB clock enable.
pselm	Output	pclk	APB select. Indicates that the slave device connected to master interface is selected and a data transfer is required.
penablem	Output	pclk	APB enable. Indicates the second and subsequent cycles of an APB transfer initiated by master interface.
pwritem	Output	pclk	APB RW transfer. Indicates an APB write access when HIGH and an APB read access when LOW.
paddrm[31:0]	Output	pclk	APB address bus.
pwdatam[31:0]	Output	pclk	APB write data. The APB master interface drives this bus.
preadym	Input	pclk	APB ready. The slave device uses this signal to extend an APB transfer.
pslverrm	Input	pclk	APB transfer error. Indicates a transfer failure. The APB peripherals are not required to support the **pslverr** pin.
prdatam[31:0]	Input	pclk	APB read data. The selected slave drives this bus during read cycles.
csysreq[a]	Input	pclk	Clock powerdown request.
csysack[a]	Output	pclk	Clock powerdown acknowledge.
cactive[a]	Output	pclk	Clock is required when driven HIGH.

a. This signal is only present if you configure this component to have an LPI.

A.2 Advanced Trace Bus interconnect signals

This section describes the following component signals and its description:

- *ATB replicator signals*.
- *ATB trace funnel signals* on page A-17.
- *ATB upsizer signals* on page A-18.
- *ATB downsizer signals* on page A-19.
- *ATB synchronous bridge signals* on page A-21.
- *ATB asynchronous bridge signals* on page A-20.

A.2.1 ATB replicator signals

Table A-12 shows the ATB replicator signals.

Table A-12 ATB replicator signals

Name	Type	Clock domain	Description
afreadys	Input	clk	ATB data flush complete on the slave port.
afvalidm0	Input	clk	ATB data flush request on the master port 0.
afvalidm1	Input	clk	ATB data flush request on the master port 1.
atbytess[<bw>:0] [a]	Input	clk	ATB number of valid bytes, LSB aligned, on the slave port.
clk	Input	clk	ATB clock.
atdatas[<dw>:0] [b]	Input	clk	ATB trace data.
atids[6:0]	Input	clk	ATB ID for current trace data.
atreadym0	Input	clk	ATB transfer ready on master port 0.
atreadym1	Input	clk	ATB transfer ready on master port 1.
resetn	Input	clk	ATB reset.
atvalids	Input	clk	ATB valid signal present.
afreadym0	Output	clk	ATB data flush complete for the master port 0.
afreadym1	Output	clk	ATB data flush complete for the master port 1.
afvalids	Output	clk	ATB data flush request for the master port.
atbytesm0[<bw>:0] [b]	Output	clk	ATB number of valid bytes, LSB aligned, on the master port.
atbytesm1[<bw>:0] [b]	Output	clk	ATB number of valid bytes, LSB aligned, on the master port.
atdatam0[<dw>:0][c]	Output	clk	ATB trace data on the master port 0.
atdatam1[<dw>:0][c]	Output	clk	ATB trace data on the master port 1.
atidm0[6:0]	Output	clk	ATB ID for current trace data on master port 0.
atidm1[6:0]	Output	clk	ATB ID for current trace data on master port 1.
atreadys	Output	clk	ATB transfer ready.
atvalidm0	Output	clk	ATB valid signal present on master port 0.
atvalidm1	Output	clk	ATB valid signal present on master port 1.

Table A-12 ATB replicator signals (continued)

Name	Type	Clock domain	Description
paddrdbg[11:2]	Input	clk	Debug APB address bus.
penabledbg	Input	clk	Debug APB enable signal, indicates second and subsequent cycles.
pseldbg	Input	clk	Debug APB component select.
pwdatadbg[31:0]	Input	clk	Debug APB write data bus.
pwritedbg	Input	clk	Debug APB write transfer.
pclkendbg	Input	clk	Debug APB clock enable.
preadydbg	Output	clk	Debug APB ready signal.
prdatadbg[31:0]	Output	clk	Debug APB read data bus.
pslverrdbg	Output	clk	Debug APB transfer error signal.
paddrdbg31	Input	clk	Enables components to distinguish between internal accesses from system software, and external accesses from a debugger.

a. <bw> has a value in the range 0-3 that is calculated at configuration time.

b. <dw> is the width of the data bus minus one.

A.2.2 ATB trace funnel signals

Table A-13 shows the ATB trace funnel signals.

Table A-13 ATB trace funnel signals

Name	Type	Clock domain	Description
afreadys<x>[a]	Input	clk	ATB data flush complete for the slave port <x>.
afvalidm	Input	clk	ATB data flush request for the master port.
atbytess<x>[<bw>:0][ab]	Input	clk	ATB number of valid bytes, LSB aligned, on the slave port <x>.
clk	Input	clk	ATB clock.
atdatas<x>[<dw>:0] [ac]	Input	clk	ATB trace data on the slave port <x>.
atids<x>[6:0] [a]	Input	clk	ATB ID for current trace data on slave port <x>.
atreadym	Input	clk	ATB transfer ready on master port.
resetn	Input	clk	ATB reset.
atvalids<x>[a]	Input	clk	ATB valid signal present on slave port <x>.
paddrdbg[11:2]	Input	clk	Debug APB address bus.
paddrdbg31	Input	clk	Enables components to distinguish between internal accesses from system software, and external accesses from a debugger.
pclkendbg	Input	clk	Debug APB clock enable.
penabledbg	Input	clk	Debug APB enable signal, indicates second and subsequent cycles.
pseldbg	Input	clk	Debug APB component select.
pwdatadbg[31:0]	Input	clk	Debug APB write data bus.

Table A-13 ATB trace funnel signals (continued)

Name	Type	Clock domain	Description
pwritedbg	Input	clk	Debug APB write transfer.
afreadym	Output	clk	ATB data flush complete for the master port.
afvalids<x>[a]	Output	clk	ATB data flush request for the slave port <x>.
atbytesm[<bw>:0][b]	Output	clk	ATB number of valid bytes, LSB aligned, on the master port.
atdatam[<dw>:0][c]	Output	clk	ATB trace data on the master port.
atidm[6:0]	Output	clk	ATB ID for current trace data on master port.
atreadys<x>[a]	Output	clk	ATB transfer ready on slave port <x>.
atvalidm	Output	clk	ATB valid signals present on master port.
prdatadbg[31:0]	Output	clk	Debug APB read data bus.
preadydbg	Output	clk	Debug APB ready signal.

a. Where the value of <x> can be 0-7.
b. Where <bw> is in the range 0-3 and is automatically calculated at configuration time.
c. Where <dw> is the data width of the interface minus one.

A.2.3 ATB upsizer signals

Table A-14 shows the ATB upsizer signals.

Table A-14 ATB upsizer signals

Signal	Type	Clock domain	Description
clk	Input	clk	Global ATB clock.
resetn	Input	clk	ATB interface reset when LOW. This signal is asserted LOW asynchronously, and deasserted HIGH synchronously.
atreadys	Output	clk	Slave is ready to accept data.
atids[6:0]	Input	clk	An ID that uniquely identifies the source of the trace.
atvalids	Input	clk	A transfer is valid during this cycle. If LOW, all other ATB signals must be ignored in this cycle.
atbytess[<sbw>:0][a]	Input	clk	The number of bytes on **ATDATA** to be captured, minus 1.
atdatas[<sdw>:0][b]	Input	clk	Trace data.
afvalids	Output	clk	This is the flush signal. All buffers must be flushed because trace capture is about to stop.
afreadys	Input	clk	This is a flush acknowledge. Asserted when buffers are flushed.
syncreqs	Output	clk	Synchronization request.
atvalidm	Output	clk	A transfer is valid during this cycle. If LOW, all the other ATB signals must be ignored in this cycle.
atreadym	Input	clk	Slave is ready to accept data.
atidm[6:0]	Output	clk	An ID that uniquely identifies the source of the trace.

Table A-14 ATB upsizer signals (continued)

Signal	Type	Clock domain	Description
afvalidm	Input	clk	This is the flush signal. All buffers must be flushed because trace capture is about to stop.
atbytesm[<mbw>:0][c]	Output	clk	The number of bytes on **ATDATA** to be captured, minus 1.
atdatam[<mdw>:0][d]	Output	clk	Trace data.
afreadym	Output	clk	This is a flush acknowledge. Asserted when buffers are flushed.
syncreqm	Input	clk	Synchronization request.

a. <sbw> has a range of 0-2.
b. <sdw> value can be either 7, 15, 31, or 63.
c. <mbw> has a range of 0-3.
d. <mdw> value can be either 7, 15, 31, 62, or 127.

A.2.4 ATB downsizer signals

Table A-15 shows the ATB downsizer signals.

Table A-15 ATB downsizer signals

Signal	Type	Clock domain	Description
clk	Input	clk	Global ATB clock.
resetn	Input	clk	The ATB interface reset. When LOW, this signal is asserted LOW, asynchronously, and deasserted HIGH synchronously.
atvalids	Input	clk	A transfer is valid during this cycle. If LOW, all the other ATB signals must be ignored in this cycle.
atreadys	Output	clk	Slave is ready to accept data.
atids[6:0]	Input	clk	An ID that uniquely identifies the source of the trace.
atbytess[sbw:0][a]	Input	clk	The number of bytes on **ATDATA** to be captured, minus 1.
atdatas[sdw-1:0][b]	Input	clk	Trace data bus.
afvalids	Output	clk	This is the flush signal. All buffers must be flushed because trace capture is about to stop.
afreadys	Input	clk	This is a flush acknowledge. Asserted when buffers are flushed.
syncreqs	Output	clk	Synchronization request.
atvalidm	Output	clk	A transfer is valid during this cycle. If LOW, all other ATB signals must be ignored in this cycle.
atreadym	Input	clk	Slave is ready to accept data.
atidm[6:0]	Output	clk	An ID that uniquely identifies the source of the trace.
atbytesm[mbw:0][c]	Output	clk	The number of bytes on **ATDATA** to be captured, minus 1.
atdatam[mdw:0][d]	Output	clk	Trace data bus.

Table A-15 ATB downsizer signals (continued)

Signal	Type	Clock domain	Description
afvalidm	Input	clk	This is the flush signal. All buffers must be flushed because trace capture is about to stop.
afreadym	Output	clk	This is a flush acknowledge. Asserted when buffers are flushed.
syncreqm	Input	clk	Synchronization request.

a. Where sbw can have the range 0-3 and is automatically calculated at configuration time based on the data width of the slave interface.
b. Where sdw is the data width of the slave interface.
c. Where mbw can have the range 0-3 and is automatically calculated at configuration time based on the data width of the master interface.
d. Where mdw is the data width of the master interface.

A.2.5 ATB asynchronous bridge signals

Table A-16 shows the ATB asynchronous bridge signals.

Table A-16 ATB asynchronous bridge signals

Name	Type	Clock domain	Description
clks	Input	clks	ATB clock.
resetsn	Input	clks	ATB reset.
clkens	Input	clks	ATB clock enable.
clkm	Input	clkm	ATB clock.
resetmn	Input	clkm	ATB reset.
clkenm	Input	clkm	ATB clock enable.
atvalids	Input	clks	ATB valid signal.
atvalidm	Output	clks	ATB valid signal.
atreadys	Output	clks	ATB transfer ready.
atreadym	Input	clkm	ATB transfer ready.
atids[6:0]	Input	clks	ATB ID for the present trace data.
atidm[6:0]	Output	clkm	ATB ID for the present trace data.
atbytess[bw:0][a]	Input	clks	ATB number of valid bytes, LSB aligned, on the slave port.
atbytesm[bw:0][a]	Output	clkm	ATB number of valid bytes, LSB aligned, on the master port.
atdatas[dw:0][b]	Input	clks	ATB trace data.
atdatam[dw:0]	Output	clkm	ATB trace data.
afvalids	Output	clks	ATB data flush request.
afvalidm	Input	clkm	ATB data flush request.
afreadys	Input	clks	ATB data flush complete.
afreadym	Input	clkm	ATB data flush complete.
syncreqs	Output	clks	Synchronization request.

Table A-16 ATB asynchronous bridge signals (continued)

Name	Type	Clock domain	Description
syncreqm	Input	clks	Synchronization request.
csysreq[c]	Input	clks	Clock powerdown request.
cactive[c]	Output	clks	Clock is required when driven HIGH.
csysack[c]	Output	clks	Clock powerdown acknowledge.

a. Where bw has a range of 0-3.
b. Where dw is either 7, 15, 31, or 63.
c. This signal is only present if you configure the device to have an LPI.

A.2.6 ATB synchronous bridge signals

Table A-17 shows the ATB synchronous bridge signals.

Table A-17 ATB synchronous bridge signals

Name	Type	Clock domain	Description
afreadys	Input	clk	ATB data flush complete.
afvalidm	Input	clk	ATB data flush request.
syncreqm	Input	clk	Synchronization request.
atbytess[<bw>:0][a]	Input	clk	ATB number of valid bytes, LSB aligned, on the slave port.
clk	Input	clk	ATB clock.
clkens	Input	clk	ATB clock enable.
clkenm	Input	clk	ATB clock enable.
atdatas[<dw>:0][b]	Input	clk	ATB trace data.
atids[6:0]	Input	clk	ATB ID for the present trace data.
atreadym	Input	clk	ATB transfer ready.
resetn	Input	clk	ATB reset for the ATCLK domain.
atvalids	Input	clk	ATB valid signal present.
afreadym	Output	clk	ATB data flush complete.
afvalids	Output	clk	ATB data flush request.
atbytesm[<bw>:0][b]	Output	clk	ATB number of valid bytes, LSB aligned, on the master port.
atdatam[<dw>:0][b]	Output	clk	ATB trace data.
atidm	Output	clk	ATB ID for current trace data.
atreadys	Output	clk	ATB transfer ready.
atvalidm	Output	clk	ATB valid signals present.

Table A-17 ATB synchronous bridge signals (continued)

Name	Type	Clock domain	Description
csysreq[c]	Input	clk	Clock powerdown request.
cactive[c]	Output	clk	Clock is required when driven HIGH.
csysack[c]	Output	clk	Clock powerdown acknowledge.

a. <bw> has a range of 0-3.

b. <dw> is the data width of the interface, minus one.

c. This signal is only present if you configure the device to have an LPI.

A.3 Timestamp component signals

This section describes timestamp component signals in the following sections:

- *Timestamp generator signals*.
- *Timestamp encoder signals* on page A-24.
- *Narrow timestamp replicator signals* on page A-25.
- *Narrow timestamp asynchronous bridge signals* on page A-25.
- *Narrow timestamp synchronous bridge signals* on page A-26.
- *Timestamp decoder signals* on page A-27.
- *Timestamp interpolator signals* on page A-28.

A.3.1 Timestamp generator signals

Table A-18 shows the timestamp generator signals.

Table A-18 Timestamp generator signals

Name	Type	Clock domain	Description
clk	Input	clk	APB clock.
resetn	Input	clk	APB reset.
tsvalueb[63:0]	Output	clk	Wide timestamp value in binary.
tsforcesync	Output	clk	Resynchronization request.
hltdbg	Input	clk	Request to halt the counter when the processor is under debug.
paddrctrl[11:2]	Input	clk	APB address.
pselctrl	Input	clk	APB select. Indicates that the slave interface is selected and a data transfer is required.
penablectrl	Input	clk	APB enable. Indicates the second and subsequent cycles of a transfer initiated on a slave interface.
pwritectrl	Input	clk	APB RW transfer. Indicates an APB write access when HIGH and an APB read access when LOW.
pwdatactrl[31:0]	Input	clk	APB write data. This bus is driven by the APB master device connected to slave interface.
preadyctrl	Output	clk	APB ready. The slave device uses this signal to extend an APB transfer.
pslverrctrl	Output	clk	Indicates a transfer failure.
prdatactrl[31:0]	Output	clk	APB read data. The slave interface drives this bus during read cycles.
paddrread[11:2]	Input	clk	APB address.
pselread	Input	clk	APB select. Indicates that the slave interface is selected and a data transfer is required.
penableread	Input	clk	APB enable. Indicates the second and subsequent cycles of a transfer initiated on a slave interface.
pwriteread	Input	clk	APB RW transfer. Indicates an APB write access when HIGH and an APB read access when LOW. Because this is an RO interface, writes have no effect.
pwdataread[31:0]	Input	clk	APB write data. The APB master device connected to slave interface drives this bus. Because this is an RO interface, writes have no effect.

Table A-18 Timestamp generator signals (continued)

Name	Type	Clock domain	Description
preadyread	Output	clk	APB ready. The slave device uses this signal to extend an APB transfer.
pslverrread	Output	clk	Indicates a transfer failure.
prdataread[31:0]	Output	clk	APB read data. Slave interface drives this bus during read cycles.

A.3.2 Timestamp encoder signals

Table A-19 shows the timestamp encoder signals.

Table A-19 Timestamp encoder signals

Name	Type	Clock domain	Description
tsclk	Input	tsclk	Timestamp clock.
tsresetn	Input	tsclk	Timestamp reset.
tsvalue[63:0]	Input	tsclk	Timestamp generator interface value.
tsforcesync	Input	tsclk	Timestamp generator interface force synchronization.
tsbit[6:0]	Output	tsclk	Timestamp encoded value.
tssync[1:0]	Output	tsclk	Timestamp synchronization bits.
tssyncready	Input	tsclk	Timestamp slave ready.

A.3.3 Narrow timestamp replicator signals

Table A-7 on page A-10 shows the narrow timestamp replicator signals.

Table A-20 Narrow timestamp replicator signals

Name	Type	Clock domain	Description
clk	Input	clk	Timestamp clock.
resetn	Input	clk	Timestamp reset.
ts_bit_valid_qualify	Input	clk	Controls if the value of **tsbit** signal is replicated or behaves similar to **tssync**.
tsbitm<x>[6:0][a]	Output	clk	Timestamp encoded value.
tssyncm<x>[1:0]	Output	clk	Timestamp synchronization bits.
tssyncreadym<x>	Input	clk	Timestamp slave ready.
tsbits[6:0]	Input	clk	Timestamp encoded value.
tssyncs[1:0]	Input	clk	Timestamp synchronization bits.
tssyncreadys	Output	clk	Timestamp slave ready.

a. Where <x> can have any value in the range 0-15.

A.3.4 Narrow timestamp asynchronous bridge signals

Table A-7 on page A-10 shows the narrow timestamp asynchronous bridge signals.

Table A-21 Narrow timestamp asynchronous bridge signals

Name	Type	Clock domain	Description
clkm	Input	clkm	Clock.
resetmn	Input	clkm	Reset.
clks	Input	clks	Clock.
resetsn	Input	clks	Reset.
tsbits[6:0]	Input	clks	Timestamp encoded value.
tssyncs[1:0]	Input	clks	Timestamp synchronization bits.
tssyncreadys	Output	clks	Timestamp slave ready.
csysreq[a]	Input	clks	Clock powerdown request.
csysack[a]	Output	clks	Clock powerdown acknowledge.
cactive[a]	Output	clks	Clock is required when driven HIGH.
tsbitm[6:0]	Output	clkm	Timestamp encoded value.
tssyncm[1:0]	Output	clkm	Timestamp synchronization bits.
tssyncreadym	Input	clkm	Timestamp slave ready.

a. This signal is only present if you configure the device to have an LPI.

A.3.5 Narrow timestamp synchronous bridge signals

Table A-7 on page A-10 shows the narrow timestamp synchronous bridge signals.

Table A-22 Narrow timestamp synchronous bridge signals

Name	Type	Clock domain	Description
clk	Input	clk	Clock.
resetn	Input	clk	Reset.
clkens	Input	clk	Clock enable.
clkenm	Input	clk	Clock enable.
tsbits[6:0]	Input	clk	Timestamp encoded value.
tssyncs[1:0]	Input	clk	Timestamp synchronization bits.
tssyncreadys	Output	clk	Timestamp slave ready.
tsbitm[6:0]	Output	clk	Timestamp encoded value.
tssyncm[1:0]	Output	clk	Timestamp synchronization bits.
tssyncreadym	Input	clk	Timestamp slave ready.
csysreq[a]	Input	clk	Clock powerdown request.
csysack[a]	Output	clk	Clock powerdown acknowledge.
cactive[a]	Output	clk	Clock is required when driven HIGH.

a. This signal is only present if you configure the device to have an LPI.

A.3.6 Timestamp decoder signals

Table A-7 on page A-10 shows the timestamp decoder signals.

Table A-23 Timestamp decoder signals

Name	Type	Clock domain	Description
clk	Input	clk	Clock.
resetn	Input	clk	Reset.
tsbit[6:0]	Input	clk	Timestamp encoded value.
tssync[1:0]	Input	clk	Timestamp synchronization bits.
tssyncready	Output	clk	Timestamp slave ready.
tsvalue[63:0]	Output	clk	Timestamp decoded value. The original value exported from the timestamp generator.

A.3.7 Timestamp interpolator signals

Table A-7 on page A-10 shows the timestamp interpolator signals.

Table A-24 Timestamp interpolator signals

Name	Type	Clock domain	Description
clk	Input	clk	Clock.
resetn	Input	clk	Reset.
tsbit[6:0]	Input	clk	Timestamp encoded value.
tssync[1:0]	Input	clk	Timestamp synchronization bits.
tsvalueb[63:0]	Input	clk	Timestamp value.
tsvalueintpb[63:0]	Output	clk	Interpolated timestamp value.

A.4 Trigger component signals

This section describes the ECT signals in the following sections:

- *Cross Trigger Interface signals*.
- *Cross Trigger Matrix signals* on page A-30.

A.4.1 Cross Trigger Interface signals

Table A-25 shows the CTI signals.

Table A-25 CTI signals

Name	Type	Clock domain	Description
cihsbypass[3:0]	Input	cticlk	Channel interface handshake bypass.
cisbypass	Input	cticlk	Channel interface sync bypass.
ctiapbsbypass	Input	cticlk	Synchronization bypass between APB and CTI clock.
ctichin[3:0]	Input	cticlk	Channel in.
ctichoutack[3:0]	Input	cticlk	Channel out acknowledge.
cticlk	Input	cticlk	CTI clock.
cticlken	Input	cticlk	CTI clock enable.
ctitrigin[7:0]	Input	cticlk	Trigger in.
ctitrigoutack[7:0]	Input	cticlk	Trigger out acknowledge.
dbgen	Input	NA	Invasive debug enable.
ctiresetn	Input	cticlk	Reset.
niden	Input	cticlk	Non-invasive debug enable.
paddrdbg[11:2]	Input	pclkdbg	Debug APB address bus.
paddrdbg31	Input	pclkdbg	Enables components to distinguish between internal accesses from system software and external accesses from a debugger.
pclkdbg	Input	pclkdbg	Debug APB clock.
pclkendbg	Input	pclkdbg	Debug APB clock enable.
penabledbg	Input	pclkdbg	Debug APB enable signal, indicates second and subsequent cycles.
presetdbgn	Input	pclkdbg	Debug APB reset.
pseldbg	Input	pclkdbg	Debug APB component select.
pwdatadbg[31:0]	Input	pclkdbg	Debug APB write data bus.
pwritedbg	Input	pclkdbg	Debug APB write transfer.
se	Input	N/A	Scan enable.
tihsbypass[7:0]	Input	cticlk	Trigger interface handshake bypass, static value.
tinidensel[7:0]	Input	cticlk	Masks when **NIDEN** is LOW, static value.
tisbypassack[7:0]	Input	cticlk	Trigger out acknowledge sync bypass, static value.
tisbypassin[7:0]	Input	cticlk	Trigger in sync bypass, static value.

Table A-25 CTI signals (continued)

Name	Type	Clock domain	Description
todbgensel[7:0]	Input	cticlk	Masks when **dbgen** is LOW, static value.
asicctl[7:0]	Output	cticlk	External multiplexer control.
ctichinack[3:0]	Output	cticlk	Channel in acknowledge.
ctichout[3:0]	Output	cticlk	Channel out.
ctitriginack[7:0]	Output	cticlk	Trigger in acknowledge.
ctitrigout[7:0]	Output	cticlk	Trigger out.
prdatadbg[31:0]	Output	pclkdbg	Debug APB read data bus.
preadydbg	Output	pclkdbg	Debug APB ready signal.

A.4.2 Cross Trigger Matrix signals

Table A-26 shows the CTM signals.

Table A-26 CTM signals

Name	Type	Clock domain	Description
cihsbypass0[CW-1:0][a]	Input	ctmclk	Handshaking bypass port 0.
cihsbypass1[CW-1:0][a]	Input	ctmclk	Handshaking bypass port 1.
cihsbypass2[CW-1:0][a]	Input	ctmclk	Handshaking bypass port 2.
cihsbypass3[CW-1:0][a]	Input	ctmclk	Handshaking bypass port 3.
cisbypass0	Input	ctmclk	Sync bypass for port 0.
cisbypass1	Input	ctmclk	Sync bypass for port 1.
cisbypass2	Input	ctmclk	Sync bypass for port 2.
cisbypass3	Input	ctmclk	Sync bypass for port 3.
ctmchin0[CW-1:0][a]	Input	ctmclk	Channel in port 0.
ctmchin1[CW-1:0][a]	Input	ctmclk	Channel in port 1.
ctmchin2[CW-1:0][a]	Input	ctmclk	Channel in port 2.
ctmchin3[CW-1:0][a]	Input	ctmclk	Channel in port 3.
ctmchoutack0[CW-1:0][a]	Input	ctmclk	Channel out acknowledge port 0.
ctmchoutack1[CW-1:0][a]	Input	ctmclk	Channel out Acknowledge port 1.
ctmchoutack2[CW-1:0][a]	Input	ctmclk	Channel out acknowledge port 2.
ctmchoutack3[CW-1:0] [a]	Input	ctmclk	Channel out acknowledge port 3.
ctmclk	Input	ctmclk	Clock.
ctmclken	Input	ctmclk	Clock enable.
ctmresetn	Input	ctmclk	Reset.

Table A-26 CTM signals (continued)

Name	Type	Clock domain	Description
se	Input	N/A	Scan enable.
ctmchinack0[CW-1:0] [a]	Output	ctmclk	Channel in acknowledge port 0.
ctmchinack1[CW-1:0] [a]	Output	ctmclk	Channel in acknowledge port 1.
ctmchinack2[CW-1:0] [a]	Output	ctmclk	Channel in acknowledge port 2.
ctmchinack3[CW-1:0] [a]	Output	ctmclk	Channel in acknowledge port 3.
ctmchout0[CW-1:0][a]	Output	ctmclk	Channel out port 0.
ctmchout1[CW-1:0][a]	Output	ctmclk	Channel out port 1.
ctmchout2[CW-1:0][a]	Output	ctmclk	Channel out port 2.
ctmchout3[CW-1:0][a]	Output	ctmclk	Channel out port 3.

a. CW is the CTM channel width.

A.5 Trace sink signals

This section describes the following component signals:

- *Trace Port Interface Unit signals*.
- *Embedded Trace Buffer signals* on page A-33.

A.5.1 Trace Port Interface Unit signals

Table A-27 shows the TPIU signals.

Table A-27 TPIU signals

name	Type	clock domain	Description
afreadys	Input	atclk	ATB data flush complete for the master port.
atbytess[1:0]	Input	atclk	ATB number of valid bytes, LSB aligned, on the slave port.
atclk	Input	atclk	ATB clock.
atclken	Input	atclk	ATB clock enable.
atdatas[31:0]	Input	atclk	ATB trace data on the slave port.
atids[6:0]	Input	atclk	ATB ID for current trace data on slave port.
atresetn	Input	atclk	ATB reset for the **atclk** domain.
atvalids	Input	atclk	ATB valid signals present on slave port.
extctlin[7:0]	Input	atclk	External control input.
flushin	Input	atclk	Flush input from the CTI.
paddrdbg[11:2]	Input	pclkdbg	Debug APB address bus.
paddrdbg31	Input	pclkdbg	Enables components to distinguish between internal accesses from system software and external accesses from a debugger.
pclkdbg	Input	pclkdbg	Debug APB clock.
pclkendbg	Input	pclkdbg	Debug APB clock enable.
penabledbg	Input	pclkdbg	Debug APB enable signal, indicates second and subsequent cycles.
presetdbgn	Input	pclkdbg	Debug APB asynchronous reset.
pseldbg	Input	pclkdbg	Debug APB component select.
pwdatadbg[31:0]	Input	pclkdbg	Debug APB write data bus.
pwritedbg	Input	pclkdbg	Debug APB write transfer.
se	Input	N/A	Scan enable.
tpctl	Input	atclk	Tie-off to report presence of **tracectl**, static value.
tpmaxdatasize[4:0]	Input	atclk	Tie-off to report maximum number of pins on **tracedata**, static value.
traceclkin	Input	traceclkin	Trace clock.
tresetn	Input	traceclkin	Trace clock asynchronous reset.
trigin	Input	atclk	Trigger input from the CTI.
afvalids	Output	atclk	ATB data flush request for the master port.

Table A-27 TPIU signals (continued)

name	Type	clock domain	Description
atreadys	Output	atclk	ATB transfer ready on slave port.
extctlout[7:0]	Output	atclk	External control output.
flushinack	Output	atclk	Flush input acknowledgement.
prdatadbg[31:0]	Output	pclkdbg	Debug APB read data bus.
preadydbg	Output	pclkdbg	Debug APB ready signal.
traceclk	Output	traceclkin	Half the frequency of the exported trace port clock, **traceclkin**.
tracectl	Output	traceclkin	Trace port control.
tracedata[31:0]	Output	traceclkin	Trace port data.
triginack	Output	atclk	Trigger input acknowledgement.

A.5.2 Embedded Trace Buffer signals

Table A-28 shows the ETB signals.

Table A-28 ETB signals

Name	Type	Clock domain	Description
afreadys	Input	atclk	ATB data flush complete.
atbytess[1:0]	Input	atclk	ATB number of valid bytes, LSB aligned, on the slave port.
atclk	Input	atclk	ATB clock.
atclken	Input	atclk	ATB clock enable.
atdatas[31:0]	Input	atclk	ATB trace data.
atids[6:0]	Input	atclk	ATB ID for the current trace data.
atresetn	Input	atclk	ATB reset for the **atclk** domain.
atvalids	Input	atclk	ATB valid signals present.
flushin	Input	atclk	Flush input from the CTI.
mbistaddr[AW-1:0]	Input	atclk	Memory BIST address.
mbistce	Input	atclk	Memory BIST chip enable.
mbistdin[31:0]	Input	atclk	Memory BIST data in.
mbistwe	Input	atclk	Memory BIST write enable.
mteston	Input	atclk	Memory BIST test is enabled.
paddrdbg[11:2]	Input	pclkdbg	Debug APB address bus.
paddrdbg31	Input	pclkdbg	Enables components to distinguish between internal accesses from system software and external accesses from a debugger.
pclkdbg	Input	pclkdbg	Debug APB clock.
pclkendbg	Input	pclkdbg	Debug APB clock enable.

Table A-28 ETB signals (continued)

Name	Type	Clock domain	Description
penabledbg	Input	pclkdbg	Debug APB enable signal. Indicates second and subsequent cycles.
presetdbgn	Input	pclkdbg	Debug APB reset.
pseldbg	Input	pclkdbg	Debug APB component select.
pwdatadbg[31:0]	Input	pclkdbg	Debug APB write data bus.
pwritedbg	Input	pclkdbg	Debug APB write transfer.
se	Input	N/A	Scan enable.
trigin	Input	atclk	Trigger input from the CTI.
acqcomp	Output	atclk	Trace acquisition complete.
afvalids	Output	atclk	ATB data flush request for the master port.
atreadys	Output	atclk	ATB transfer ready on slave port.
flushinack	Output	atclk	Flush input acknowledgement.
full	Output	atclk	The cxetb RAM overflow or wrapped around.
mbistdout[31:0]	Output	atclk	Memory BIST data out.
prdatadbg[31:0]	Output	pclkdbg	Debug APB read data bus.
preadydbg	Output	pclkdbg	Debug APB ready signal. Use this signal to extend an APB transfer.
triginack	Output	atclk	Trigger input acknowledgement.

A.6 Authentication and event bridges

This section describes the following component signals:

- *Authentication asynchronous bridge signals*.
- *Authentication synchronous bridge signals* on page A-36.
- *Event asynchronous bridge signals* on page A-36.

A.6.1 Authentication asynchronous bridge signals

Table A-29 shows the authentication asynchronous bridge signals.

Note

The master clock is the domain that drives the slave interface of this bridge.

Table A-29 Authentication asynchronous bridge signals

Name	Type	Clock domain	Description
clks	Input	clks	Authentication clock.
clkm	Input	clkm	Authentication clock.
resetsn	Input	clks	Authentication reset.
resetmn	Input	clkm	Authentication reset.
dbgens	Input	clks	Invasive debug enable.
nidens	Input	clks	Non-invasive debug enable.
spidens	Input	clks	Secure invasive debug enable.
spnidens	Input	clks	Secure non-invasive debug enable.
dbgswens	Input	clks	Invasive software debug enable.
dbgenm	Output	clkm	Invasive debug enable.
nidenm	Output	clkm	Non-invasive debug enable.
spidenm	Output	clkm	Secure invasive debug enable.
spnidenm	Output	clkm	Secure non-invasive debug enable.
dbgswenm	Output	clkm	Invasive software debug enable.

A.6.2 Authentication synchronous bridge signals

Table A-30 shows the authentication synchronous bridge signals.

Table A-30 Authentication synchronous bridge signals

Name	Type	Clock domain	Description
clk	Input	clk	Authentication clock.
resetn	Input	clk	Authentication reset.
dbgens	Input	clk	Invasive debug enable.
nidens	Input	clk	Non-invasive debug enable.
spidens	Input	clk	Secure invasive debug enable
dbgswens	Input	clk	Invasive software debug enable.
dbgenm	Output	clk	Invasive debug enable.
nidenm	Output	clk	Non invasive debug enable.
spidenm	Output	clk	Secure invasive debug enable.
spnidenm	Output	clk	Secure non-invasive debug enable.
dbgswenm	Output	clk	Invasive software debug enable.
spnidens	Input	clk	Secure non-invasive debug enable.

A.6.3 Event asynchronous bridge signals

Table A-31 shows the event asynchronous bridge signals.

Note

Master clock is the domain that drives the slave interface to this bridge.

Table A-31 Event asynchronous bridge signals

Name	Type	Clock domain	Description
clks	Input	clks	Clock.
clkens	Input	clks	Clock enable.
resetsn	Input	clks	Reset.
clkm	Input	clkm	Clock.
clkenm	Input	clkm	Clock enable.
resetmn	Input	clkm	Reset.
events	Input	clks	Event request.
eventacks	Output	clks	Event acknowledge.
eventm	Output	clkm	Event request.
eventackm	Input	clkm	Event acknowledge.

A.7 Granular power requestor signals

Table A-7 on page A-10 shows the granular power requestor signals.

Table A-32 Granular power requestor signals

Name	Type	Clock domain	Description
clk	Input	clk	Clock
resetn	Input	clk	Reset
cpwrupreq[31:0]	Output	clk	Powerup request. This signal requests the system power controller to: • Powerup the target power domain. • Enable the clocks. De-asserting **cpwrupreq** issues a request to the system power controller to remove power to a domain.
cpwrupack[31:0]	Input	clk	Powerup acknowledge. This signal acknowledges that a system power controller responded to a powerup or powerdown request.
pseldbg	Input	clk	DAP select.
penabledbg	Input	clk	DAP enable.
pwritedbg	Input	clk	DAP write or read.
paddrdbg[11:2]	Input	clk	DAP compressed address bus.
pwdatadbg[31:0]	Input	clk	DAP write data bus.
preadydbg	Output	clk	DAP ready.
pslverrdbg	Output	clk	DAP slave error.
prdatadbg[31:0]	Output	clk	DAP compressed address bus.
paddr31dbg	Input	clk	Enables components to distinguish between internal accesses from system software and external accesses from a debugger.

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
Correction to signal name capitalization and signal directions.	Chapter 2 *Functional Overview*	All revisions
Correction to signal name capitalization and signal directions.	Appendix A *Signal Descriptions*	All revisions
Clarification of management register description.	Chapter 3 *Programmers Model*	All revisions
Component revision updated in the identification registers.	Chapter 3 *Programmers Model*	r1p0
ATB replicator IDFILTER0 diagram updated.	Figure 3-57 on page 3-59	All revisions
Moved and updated DAPBUS interconnect and APB interconnect and ROM table chapters into subsections of the Debug Access Port.	Chapter 4 *Debug Access Port*	All revisions
Component versions updated in block summary.	*CoreSight SoC block summary* on page 1-4	r1p0 and above
Detail added on clock domain crossing bridges.	Chapter 4 *Debug Access Port*	All revisions
Detail added on clock domain crossing bridges.	Chapter 5 *ATB Interconnect Components*	All revisions

Table B-2 Differences between Issue A and Issue B (continued)

Change	Location	Affects
Detail added on clock domain crossing bridges.	Chapter 6 *Timestamp Components*	All revisions
Event Asynchronous Bridge component information included.	Entire document	All revisions
Granular Power Requestor component added and referenced in *Granular Power Requestor* on page 2-24 and *Granular power requestor signals* on page A-37.	Chapter 10 *Granular Power Requestor*	r1p0 and above
Timestamp interpolator component added and referenced in *Timestamp interpolator* on page 2-17 and *Timestamp interpolator signals* on page A-28.	*Timestamp interpolator* on page 6-12	r1p0 and above

Table B-3 Differences between Issue B and Issue C

Change	Location	Affects
Updated *Structure of CoreSight SoC* on page 1-2 section.	Chapter 1 *Introduction*	All revisions
Updated Narrow timestamp asynchronous bridge revision in *CoreSight SoC block summary* on page 1-4.	Chapter 1 *Introduction*	All revisions
Updated *Product revisions* on page 1-7 for r2p0.	Chapter 1 *Introduction*	r2p0
Added **rombaseaddrl[31:0]** and **rombaseaddru[31:0]** to Figure 2-6 on page 2-5.	Chapter 2 *Functional Overview*	All revisions
Added **rombaseaddr[31:0]** to Figure 2-7 on page 2-6.	Chapter 2 *Functional Overview*	All revisions
Updated *Event asynchronous bridge* on page 2-23 section.	Chapter 2 *Functional Overview*	All revisions
Moved and updated *JTAG-DP register summary* on page 3-182 into subsection of the *Debug port register summary* on page 3-182.	Chapter 3 *Programmers Model*	All revisions
Moved and updated *JTAP-DP register descriptions* on page 3-209 into subsection of the *Debug port implementation-specific registers* on page 3-200.	Chapter 3 *Programmers Model*	All revisions
Reset value correction in *JTAG-DP register summary* on page 3-182.	Chapter 3 *Programmers Model*	All revisions
Updated the description in Table 3-225 on page 3-189.	Chapter 3 *Programmers Model*	All revisions
Updated the description in Table 3-235 on page 3-197.	Chapter 3 *Programmers Model*	All revisions
Component revision updated in the identification registers.	Chapter 3 *Programmers Model*	r2p0
Updated *DAP flow of control* on page 4-5 section.	Chapter 4 *Debug Access Port*	All revisions
Moved and updated *Operation in JTAG-DP mode* on page 4-8 and *Operation in SW-DP mode* on page 4-9 into subsection of the *JTAG and SWD interface* on page 4-8.	Chapter 4 *Debug Access Port*	All revisions
Updated Table 5-4 on page 5-13 in *ATB upsizer* on page 5-12 section.	Chapter 5 *ATB Interconnect Components*	All revisions
Updated *Arbitration* on page 5-4 section in *ATB funnel* on page 5-4.	Chapter 5 *ATB Interconnect Components*	All revisions
Added **rombaseaddrl[31:0]** and **rombaseaddru[31:0]** to Table A-6 on page A-8.	Appendix A *Signal Descriptions*	All revisions
Added **rombaseaddr[31:0]** to Table A-7 on page A-10.	Appendix A *Signal Descriptions*	All revisions

Table B-4 Differences between Issue C and Issue D

Change	Location	Affects
Updated the signal case for the following block diagrams: • Figure 2-25 on page 2-18. • Figure 2-26 on page 2-19. • Figure 2-27 on page 2-20. • Figure 2-28 on page 2-21.	Chapter 2 *Functional Overview*	All
Updated the offset value for CIDR 0-3 in Table 3-251 on page 3-210	Chapter 3 *Programmers Model*	All
Updated the top-level signal case for ECT components.	Chapter 7 *Embedded Cross Trigger*	All
Updated the top-level signal case for TPIU components.	Chapter 8 *Trace Port Interface Unit*	All
Updated the top-level signal case for ETB components.	Chapter 9 *Embedded Trace Buffer*	All
Updated the top-level signal case for ECT, TPIU, and ETB components.	Appendix A *Signal Descriptions*	All
Updated the component version references	Table 1-1 on page 1-4 Table 3-54 on page 3-54	All