CoreSight™ TPIU-Lite

Revision: r0p0

Technical Reference Manual

ARM DDI 0317A

CoreSight TPIU-Lite
Technical Reference Manual

Release Information

The following changes have been made to this book.

Change History

Date	Issue	Confidentiality	Change
6 January 2006	A	Non-confidential	First release for r0p0

Proprietary Notice

Confidentiality Status

Product Status

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

Web Address

`http://www.arm.com`

Contents

CoreSight TPIU-Lite Technical Reference Manual

List of Tables
CoreSight TPIU-Lite Technical Reference Manual

List of Figures
CoreSight TPIU-Lite Technical Reference Manual

Preface

This preface introduces the *CoreSight TPIU-Lite r0p0 Technical Reference Manual*. It contains the following sections:

- *About this manual* on page x
- *Feedback* on page xiv.

About this manual

This is the *Technical Reference Manual* (TRM) for the CoreSight *Trace Port Interface Unit Lite* (TPIU-Lite).

Product revision status

The r*n*p*n* identifier indicates the revision status of the product described in this manual, 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 manual is written for the following target audience:

- Hardware and software engineers who want to incorporate a TPIU-Lite into their design and output real-time trace information from an ASIC.
- Software engineers writing tools to use a TPIU-Lite component.

This manual assumes that readers are familiar with AMBA bus design and trace protocol.

Using this manual

This manual is organized into the following chapters:

Chapter 1 *Introduction*

Read this chapter for a high-level view of the TPIU-Lite and a description of its features.

Chapter 2 *Functional Overview*

Read this chapter for a description of the functions of the TPIU-Lite and how they operate.

Chapter 3 *Programmer’s Model*

Read this chapter for description of the TPIU-Lite registers.

Appendix A *TPIU-Lite Ports*

Read this chapter for a description of the TPIU-Lite input and output signals.

Glossary Read the Glossary for definitions of terms used in this manual.

Conventions

Conventions that this manual can use are described in:

- *Typographical*
- *Timing diagrams*
- *Signals* on page xii
- *Numbering* on page xiii.

Typographical

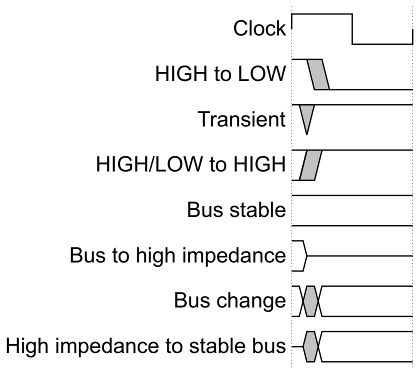
The typographical conventions are:

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

Timing diagrams

The figure named *Key to timing diagram conventions* on page xii 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 and LOW for active-LOW signals.
Lower-case n	Denotes an active-LOW signal.
Prefix A	Denotes global *Advanced eXtensible Interface* (AXI) signals:
Prefix AR	Denotes AXI read address channel signals.
Prefix AW	Denotes AXI write address channel signals.
Prefix B	Denotes AXI write response channel signals.
Prefix C	Denotes AXI low-power interface signals.
Prefix H	Denotes *Advanced High-performance Bus* (AHB) signals.
Prefix P	Denotes Advanced Peripheral Bus (APB) signals.
Prefix R	Denotes AXI read data channel signals.
Prefix W	Denotes AXI write data channel signals.

Numbering

The numbering convention is:

<size in bits>'<base><number>

This is a Verilog method of abbreviating constant numbers. For example:

- 'h7B4 is an unsized hexadecimal value.
- 'o7654 is an unsized octal value.
- 8'd9 is an eight-bit wide decimal value of 9.
- 8'h3F is an eight-bit wide hexadecimal value of `0x3F`. This is equivalent to b00111111.
- 8'b1111 is an eight-bit wide binary value of b00001111.

Further reading

This section lists publications by ARM Limited, and by third parties.

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

ARM publications

This manual contains information that is specific to the TPIU-Lite. See the following documents for other relevant information:

- *CoreSight System Design Guide*, ARM DGI 0012
- *CoreSight Architecture Specification*, ARM IHI 0029
- *CoreSight Design Kit Technical Reference Manual*, ARM DDI 0314
- *ETM Architecture Specification*, ARM IHI 0014
- *AMBA® 3 APB Protocol*, ARM IHI 0024
- *RealView ICE User Guide*, ARM DUI 0155.

Feedback

ARM Limited welcomes feedback on the TPIU-Lite and its documentation.

Feedback on this product

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

- the product name
- a concise explanation of your comments.

Feedback on this manual

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

- the title
- the number
- the relevant page number(s) to which your comments apply
- a concise explanation of your comments.

ARM Limited also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter introduces the CoreSight TPIU-Lite. It contains the following section:

1.1 About the TPIU-Lite

The TPIU-Lite acts as a bridge between on-chip trace data and *Application Specific Integrated Circuit* (ASIC) pads. The trace data is then captured by a *Trace Port Analyzer* (TPA). You can use the TPIU-Lite to control output port widths and to manually flush data. The TPIU-Lite enables tracing of *Embedded Trace Macrocells* (ETMs) using an *Advanced Trace Bus* (ATB) input and synchronous Trace Port.

The TPIU-Lite contains the following components:

- ATB interface
- *Advanced Peripheral Bus* (APB) interface
- buffer
- register bank
- trace out serializer.

——— **Note** ———

The TPIU-Lite is a version of the TPIU supplied with the CoreSight Design Kit, with fewer features. It does not contain the following blocks:

- A formatter, because only one trace source can be connected to the component. There is no provision for tracing from multiple sources.
- A FIFO. When data is still being serialized in the component, the ATB interface stalls until another packet can be accepted.
- A pattern generator. This functionality is not included in the TPIU-Lite.

For more information on the additional features and functionality provided by the TPIU, see the *CoreSight Design Kit Technical Reference Manual*.

Chapter 2
Functional Overview

This chapter describes the major components of the CoreSight TPIU-Lite, and how they operate. It contains the following sections:

2.1 TPIU-Lite structure

Figure 2-1 shows the structure of the TPIU-Lite.

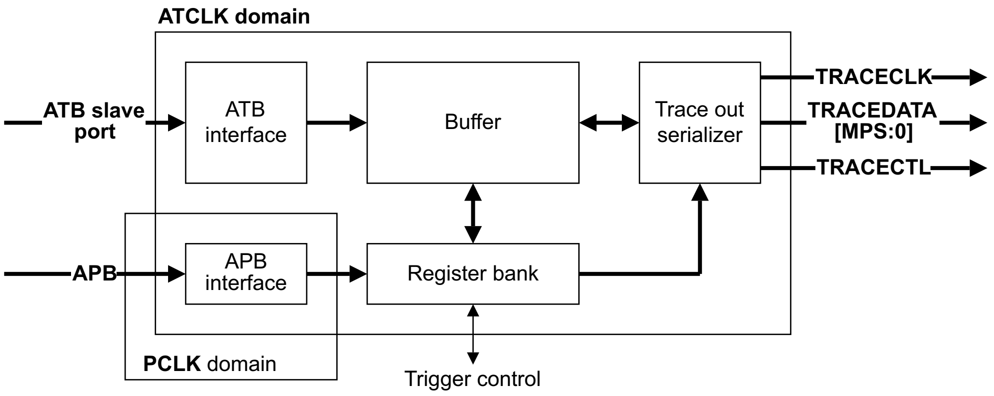


Figure 2-1 TPIU-Lite structure

The TPIU-Lite comprises the following interface blocks:

- *Advanced Trace Bus* (ATB) interface
- *Advanced Peripheral Bus* (APB) interface
- buffer
- register bank
- trace out.

The behavior of the blocks is as follows:

ATB interface

The TPIU-Lite accepts trace data direct from a trace source.

APB interface

The APB interface is the programming interface for the TPIU-Lite.

Buffer Holds transactions until enough data has been gathered for the trace out block.

Register bank

Contains the management, control and status registers for triggers and flushing behavior.

Trace out The trace out port serializes data before it goes off-chip.

2.2 Trace out port

The trace out port is described in:

- *Synchronous trace port*
- *Synchronous encoding pin protocol* on page 2-4
- *Constraining the supported TRACEDATA port widths* on page 2-4.

2.2.1 Synchronous trace port

The TPIU is configured to allow for the largest port size allowable, 32 bits of **TRACEDATA**, **TRACECTL**, and **TRACECLK**.

- **TRACECLK** is always exported to enable synchronization back with the data.
- **TRACECTL** is always required to indicate valid, non-valid and trigger cycles.
- **TRACEDATA** can be defined in various sizes up to 32 bits. For backwards compatibility and usage with ETMv3 trace capture devices, a minimum port width of 2 is permitted, that is, **TRACEDATA[1:0]**.

Table 2-1 shows some example port configurations.

Table 2-1 Example port configurations

TRACECLK present	TRACECTL present	TRACEDATA width	Total pin count	Comment
Yes	Yes	32 bits [31:0]	34	Largest implementation
Yes	Yes	8 bits [7:0]	10	Typical ETM-compatible TPA implementation
Yes	Yes	2 bits [1:0]	4	Smallest implementation with typical TPAs

2.2.2 Synchronous encoding pin protocol

Any attached capture device is expected to store the data as defined as valid in Table 2-2. Data is unmodified from the originating trace source.

Table 2-2 Synchronous trace port pin encoding

Pin	Logic 0	Logic 1	Idle State (no data)	Valid Data
TRACECLK	LOW	HIGH	Not applicable, constant clock	Not applicable
TRACECTL	LOW[a]	HIGH[a]	HIGH[a]	LOW[a]
TRACEDATA[..]	LOW[a]	HIGH[a]	**TRACEDATA[0]** is HIGH[a]	Number of valid bits controlled by CPR (`0x004`)[a]

a. Sampled on **TRACECLK** transition, HIGH-LOW and LOW-HIGH.

2.2.3 Constraining the supported TRACEDATA port widths

When placing on an *Application Specific Integrated Circuit* (ASIC), not all the signals of **TRACEDATA** go to pads, that is, only **TRACEDATA**[(MDS-1):0] go to pins. If *Maximum Data Size* (MDS) is not 32, that is, the maximum supported data width for capture by a *Trace Port Analyzer* (TPA) is less than 32 bits of data, this must be reflected in the programmer's view of the Supported Port Size Register through the tie-off input **TPMAXDATASIZE[4:0]**. See *Supported Port Size Register, 0x000* on page 3-5 for more details.

The tie-off must be set to represent the number of **TRACEDATA** connections that go to ASIC pads, with all LOW indicating 1 pin, which is the minimum possible, and all HIGH indicating 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`.

2.3 Triggers

The TPIU-Lite can only indicate triggers from either the completion of a flush request or directly from the activation of the **TRIGIN** input. If neither of these control bits in the Formatter and Flush Control Register is set, no triggers are indicated on the Trace Out port. See the *CoreSight Architecture Specification* for more information on triggers.

2.3.1 Output indication

The only indication of triggers is through the synchronous trace port, **TRACECTL**. Table 2-3 shows the synchronous pin protocol encodings.

Table 2-3 Synchronous pin protocol

TRACECTL	TRACEDATA		Description	Trigger	Capture
	[0]	[1]			
0	x	x	Normal trace data	No	Yes
1	0	1	Trigger	Yes	No
1	1	x	TraceDisable	No	No

2.3.2 Correlation with AFVALID

When a trigger signal is received by the TPIU-Lite, 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 that could relate to the trigger is output before the TPA or other capture device is stopped, by enabling both the Trigger on Flush Completion and Flush on Trigger Event bits, [10] and [5] respectively, in the Formatter and Flush Control Register.

2.4 Flushing

There is limited support for flushing in the TPIU-Lite. Flushing can be directly requested by setting bit [6], or by enabling bit [5] (FOnTrig) in the Formatter and Flush Control Register. While a flush operation is being processed, bit [0] (FlInProg) in the Formatter and Flush Status Register is HIGH. See *Formatter and Flush Status Register, 0x300* on page 3-6. If both a manual flush and a Trigger Event flush are active at the same time, two flushes occur. The highest priority is a Flush on Trigger Event if they occur simultaneously. See the *CoreSight Architecture Specification* for more information on flushing.

2.5 Stopping trace

Although trace data is unmodified, when the end of trace session occurs and the TPIU-Lite is set to Stop, there is an extra sequence added to the end of the data stream to enable tools to discover all the trace data to decompress. This sequence is shown in Figure 2-2 and is always added when the TPIU-Lite is stopping.

31 24	23 16	15 8	7 0
0xaa [Real data]	0x55 [Real data]	0xaa [Real data]	0x55 [Real data]
0x01	0x55 [Real data]	0xaa [Real data]	0x55 [Real data]
0x00	0x00	0x00	0x00
0x00	0x00	0x00	0x00

Figure 2-2 End of sequence pattern example

Before decompression of any captured trace, tools must remove the end pattern even if it might not have been captured. For example, if the trace capture device stopped early after observing a trigger, the stored trace might not contain the end of sequence pattern. However, the tools must attempt to remove a similar sequence to guarantee safe decompression.

Figure 2-3 shows a bit-stream output from a trace port with only the first section stored in a capture device. This is a stream of data packets from the trace source.

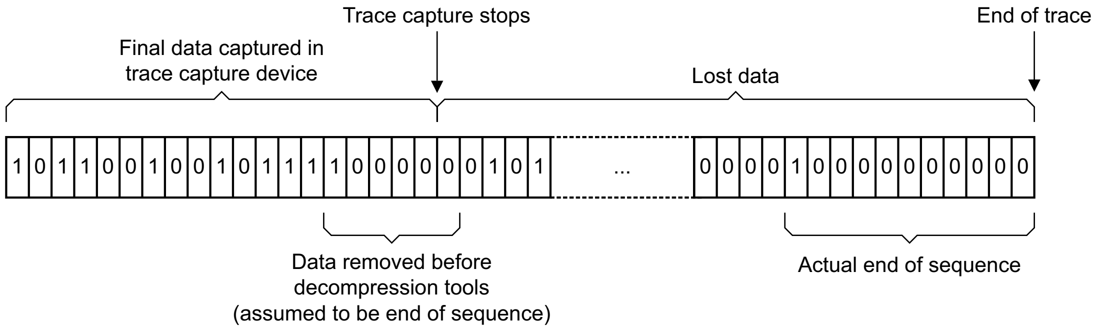


Figure 2-3 Trace stream showing lost data

2.6 Bypass mode

This is the only mode of operation the TPIU-Lite supports. This mode passes the **ATDATA** values from the input to the output with no modifications of the data stream. It is a requirement of the trace source protocol to have bit or byte alignment and synchronization because this cannot be inserted without an understanding of the trace protocol. See the documentation for the trace source for information on trace source protocol.

2.7 TRACECLK generation

In an ideal environment **TRACECLK** is derived from the negative edge of **ATCLK** to create a sample point within the centre of the stable data, **TRACEDATA** and **TRACECTL**, on each changing edge of **TRACECLK** irrespective of the operating frequency. This method creates a large number of issues during clock-tree synthesis, layout, and static timing analysis.

TRACECLK is a divided by two, exported version of **ATCLK**. The reason for creating a half clock is that the limiting factor for the Trace Out port is the slew rate from a zero-to-one and one-to-zero. If it is possible to detect logic 1 and logic 0 on the exported clock within one cycle, it is also possible to detect two different values on the exported data pins.

There is no requirement to either invert the clock or use negative-edge registers in the generation of **TRACECLK**. The register that creates the divided by two clock is a standard positive-edge register that operates synchronously to the **TRACEDATA** and **TRACECTL** registers. This method simplifies synthesis in the early stages and ensures when clock tree synthesis is performed, all the registers are operating at the same time. To create the sample point at a stable point in the exported data, a delay must be added to the path of **TRACECLK** between the register and the pad.

Figure 2-4 on page 2-10 shows **TRACECLK** at different points in the design and its relationship to the data and control signals, **TRACEDATA** and **TRACECTL**. At the moment of creation from the final registers of the Trace Out port signals, all data edges are aligned as shown at point A in Figure 2-4 on page 2-10.

All the signal paths to the pads are subject to delays as a result of the differing path lengths at point B from wire delay. These delays must be minimized where possible by placing the registers as close to the pads as possible. Each path must be re-balanced to remove the relative skew between signals by adding in equivalent delays. An extra delay must be incorporated on the **TRACECLK** path to ensure the waveform at point C is achieved and that the rising and falling edges of **TRACECLK** correspond to the center of stable data on **TRACECTL** and **TRACEDATA** as shown in Figure 2-5 on page 2-11.

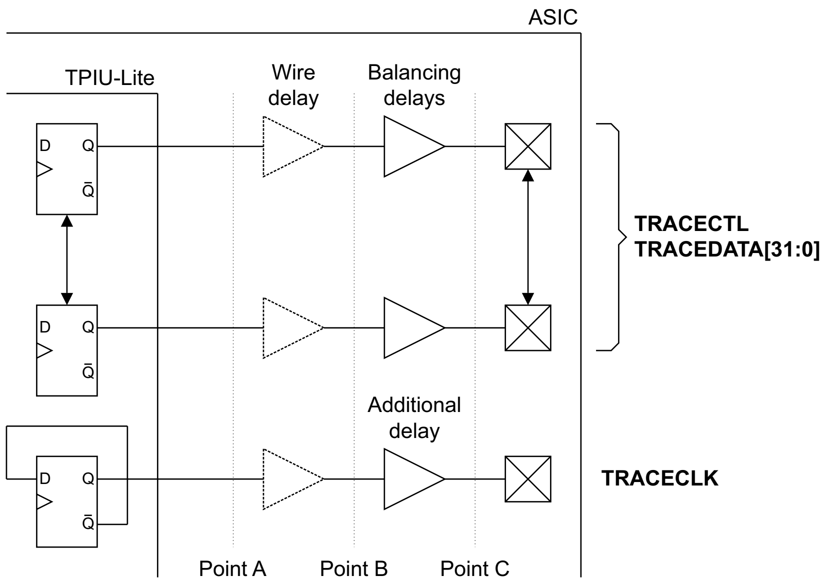


Figure 2-4 Paths of TRACECLK, TRACEDATA, and TRACECTL to pads

Figure 2-5 on page 2-11 shows how the rising and falling edges of **TRACECLK** correspond to the center of stable data on **TRACECTL** and **TRACEDATA** at point C.

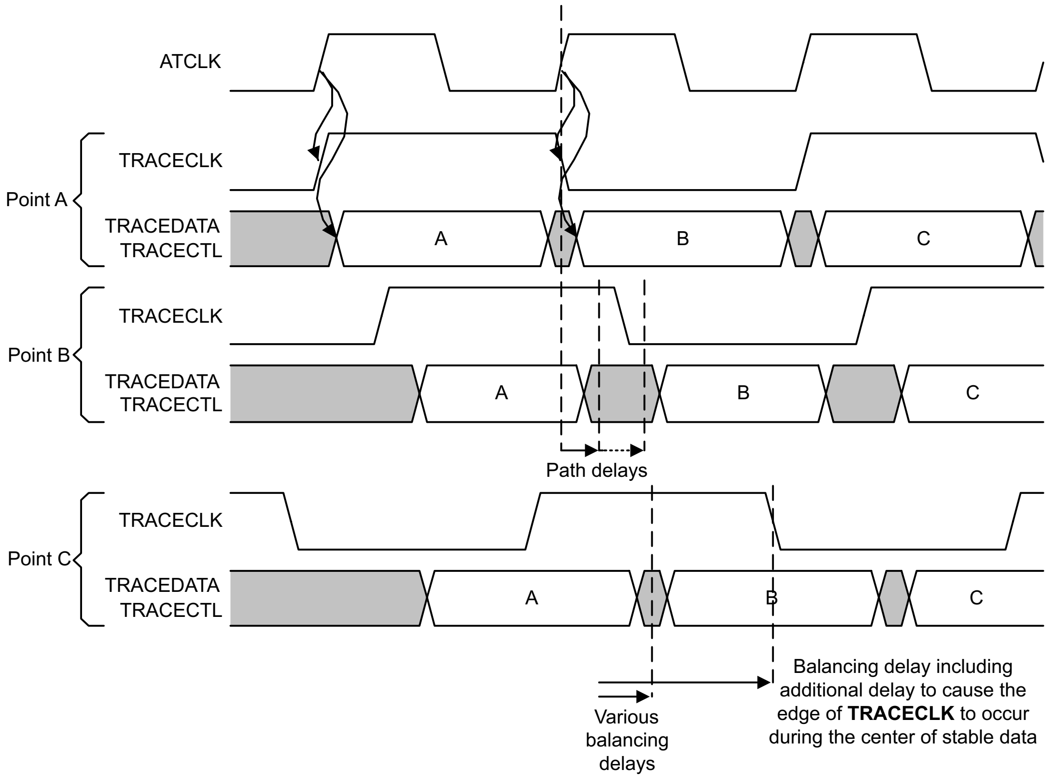


Figure 2-5 TRACECLK timing in relation to TRACEDATA and TRACECTL

2.8 Authentication requirements

The TPIU-Lite does not require any authentication signals capable of disabling it.

Chapter 3
Programmer's Model

This chapter describes the CoreSight TPIU-Lite programmer's model. It contains the following sections:

3.1 About the programmer's model

The TPIU-Lite has a 4KB location block in the memory map. The base address of the registers is not fixed but the address offsets are fixed.

The following apply to all TPIU-Lite registers:

- All registers are word-aligned and must be accessed as 32-bit.
- Reserved or unused address locations must not be accessed because this can result in Unpredictable behavior.
- Reserved or unused bits of registers must be written as zero, and ignored on read unless otherwise stated in the relevant text.
- All register bits are reset to a logic 0 by a system or power-on reset unless otherwise stated in the relevant text.
- All registers support read and write (R/W) accesses unless otherwise stated in the relevant text. A write (WO) updates the contents of a register and a read (RO) returns the contents of the register.

3.2 Register summary

This section defines all the visible registers that can be accessed from the APB interface. Table 3-1 shows the TPIU programmable registers.

Table 3-1 TPIU programmable registers

Offset	Type	Width	Reset value	Name	Description
`0x000`	RO	32	`0x8000808A`	Supported port size	See *Trace port control registers* on page 3-5
`0x004`	R/W	32	`0x00000002`	Current port size	
`0x300`	RO	4	`0x6`	Formatter and Flush Status	See *Formatter and flush registers* on page 3-6
`0x304`	R/W	14	`0x1000`	Formatter and Flush Control	
`0x308`	R/W	12	`0x0`	Formatter Synchronization Counter	
`0xEE4`	WO	1	-	Integration Register, ITTRFLINACK	See *Integration test registers* on page 3-10
`0xEE8`	RO	1	-	Integration Register, ITTRFLIN	
`0xEEC`	RO	5	-	Integration Register, ITATBDATA0	
`0xEF0`	WO	2	-	Integration Register, ITATBCTR2	
`0xEF8`	RO	10	-	Integration Register, ITATBCTR0	
`0xF00`	R/W	1	`0x0`	Integration Mode Control Register	See *CoreSight-defined registers* on page 3-14
`0xFA0`	R/W	4	0xF	Claim Tag Set	
`0xFA4`	R/W	4	0x0	Claim Tag Clear	
`0xFB0`	WO	32	-	Lock Access	
0xFB4	RO	3	`0x3/0x0`	Lock Status	
`0xFB8`	RO	8	0x00	Authentication Status	
0xFC8	RO	13	0x0000	Device ID	
`0xFCC`	RO	8	`0x11`	Device Type Identifier	
`0xFD0`	RO	8	`0x04`	Peripheral ID4	
`0xFD4`	RO	8	-	Peripheral ID5 (unused)	
`0xFD8`	RO	8	-	Peripheral ID6 (unused)	
`0xFDC`	RO	8	-	Peripheral ID7 (unused)	

Table 3-1 TPIU programmable registers (continued)

Offset	Type	Width	Reset value	Name	Description
`0xFE0`	RO	8	`0x41`	Peripheral ID0	See *CoreSight-defined registers* on page 3-14
`0xFE4`	RO	8	`0xB9`	Peripheral ID1	
`0xFE8`	RO	8	`0x0B`	Peripheral ID2	
`0xFEC`	RO	8	`0x00`	Peripheral ID3	
`0xFF0`	RO	8	`0x0D`	Component ID0	
`0xFF4`	RO	8	`0x90`	Component ID1	
`0xFF8`	RO	8	`0x05`	Component ID2	
`0xFFC`	RO	8	`0xB1`	Component ID3	

3.3 Trace port control registers

This section describes the trace port control registers:

- *Supported Port Size Register, 0x000*
- *Current Port Size Register, 0x004*.

3.3.1 Supported Port Size Register, 0x000

The TPIU-Lite only supports port sizes for **TRACEDATA** of 32, 16, 8, 4, and 2 bits. This value can be changed if fewer pins are taken to the output pins as indicated in **TPMAXDATASIZE**. Figure 3-1 shows the bit assignments.

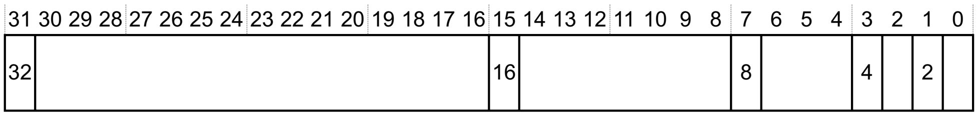


Figure 3-1 Supported Port Size Register bit assignments

3.3.2 Current Port Size Register, 0x004

This register has the same format as the Supported Port Size Register but only one bit is set. All other bits 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 is the smallest port width allowable in the Supported Port Size register. `0x00000002` represents a 2-bit **TRACEDATA**.

3.4 Formatter and flush registers

This section describes the formatter and flush registers:

- *Formatter and Flush Status Register, 0x300*
- *Formatter and Flush Control Register, 0x304* on page 3-7
- *Formatter Synchronization Counter Register, 0x308* on page 3-8.

For more information on flushing, see *Flushing* on page 2-6.

3.4.1 Formatter and Flush Status Register, 0x300

The Formatter and Flush Status Register indicates the implemented Trigger Counter multipliers and other supported features of the trigger system. Figure 3-2 shows the bit assignments.

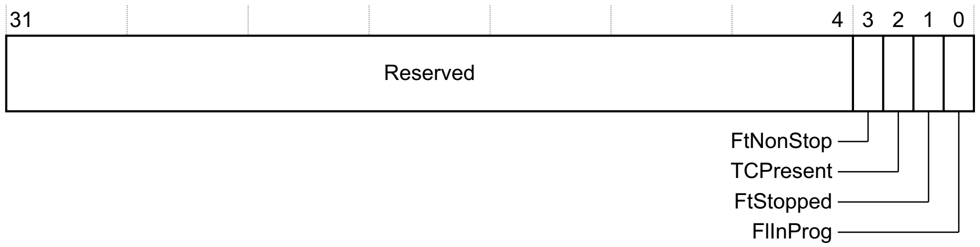


Figure 3-2 Formatter and Flush Status Register bit assignments

Table 3-2 shows the bit assignments.

Table 3-2 Formatter and Flush Status Register bit assignments

Bits	Name	Description
[31:4]	-	Reserved. RAZ on reads, SBZP on writes.
[3]	FtNonStop	Reads LOW indicating the formatter can be stopped. Settings can only be altered with the formatter stopped.
[2]	TCPresent	Always HIGH. **TRACECTL** must always be present on ASIC pads.
[1]	FtStopped	Formatter stopped. This reads LOW when the Trace Port is still active. From reset, this is HIGH indicating that the TPIU-Lite is stopped. To start it, clear bit[12] of the Formatter and Flush Control Register.
[0]	FlInProg	Flush In Progress. This is an indication of the current state of **AFVALID**.

3.4.2 Formatter and Flush Control Register, 0x304

The Formatter and Flush Control Register controls the generation of stop, trigger, and flush events. Figure 3-3 shows the bit assignments.

Note

Only bits 12, 10, 8, 6, and 5 are writable. All other bits are read as zero and cannot be modified.

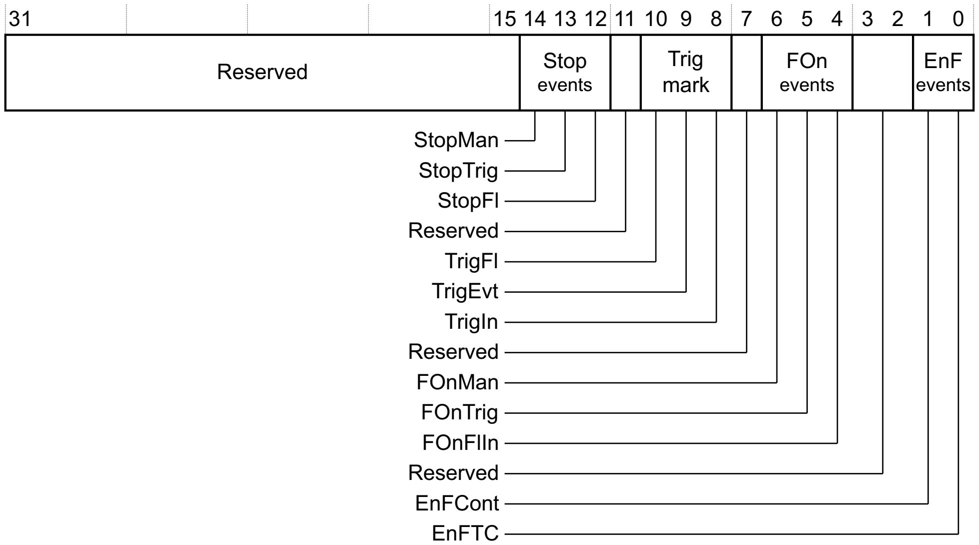


Figure 3-3 Formatter and Flush Control Register bit assignments

Table 3-3 shows the bit assignments.

Table 3-3 Formatter and Flush Control Register bit assignments

Bits	Name	Description
[31:15]	-	Reserved. RAZ on reads, SBZP on writes.
[14]	StopMan	Manually stop the output of trace.
[13]	StopTrig	Stop the formatter after a Trigger Event[a] is observed. Reset to disabled, or zero.
[12]	StopFl	Stop the formatter after a flush completes (return of **AFREADY**). This forces the FIFO to drain off any part-completed packets. Setting this bit enables this function which is set on reset, to enable the TPIU-Lite to temporarily clear this bit.

Table 3-3 Formatter and Flush Control Register bit assignments (continued)

Bits	Name	Description
[11]	-	Reserved. RAZ on reads, SBZP on writes.
[10]	TrigFl	Indicates a trigger on Flush completion on **AFREADY** being returned.
[9]	TrigEvt	Indicates a trigger on a Trigger Event[a].
[8]	TrigIn	Indicates a trigger on **TRIGIN** being asserted[b].
[7]	-	Reserved. RAZ on reads, SBZP on writes.
[6]	FOnMan	Manually generate a flush of the system. Setting this bit causes a flush to be generated. This is cleared when this flush has been serviced. This bit is clear on reset.
[5]	FOnTrig	Generate flush using Trigger event. Set this bit to cause a flush of data in the system when a Trigger Event[a] occurs. Reset value is this bit clear.
[4]	FOnFlIn	Generate flush using the **FLUSHIN** interface. Set this bit to enable use of the **FLUSHIN** connection. This is clear on reset.
[3:2]	-	Reserved. RAZ on reads, SBZP on writes.
[1]	EnFCont	Continuous Formatting, data always present. This bit is always LOW because the formatting mode cannot be altered.
[0]	EnFTC	Enable Formatting. This bit is always LOW because the formatting mode cannot be altered.

a. A Trigger Event is defined as when the Trigger counter reaches zero, where fitted or, in the case of the trigger counter being zero, or not fitted, when **TRIGIN** is HIGH. The TPIU-Lite does not implement a Trigger Counter so a Trigger Event is always based on the rising edge of **TRIGIN**. For more information on triggers, see *Triggers* on page 2-5.

b. Holding **TRIGIN** HIGH does not result in multiple triggers being indicated.

3.4.3 Formatter Synchronization Counter Register, 0x308

The Formatter Synchronization Counter Register enables effective use on different sized *Trace Port Analyzers* (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. Figure 3-4 on page 3-9 shows the bit assignments.

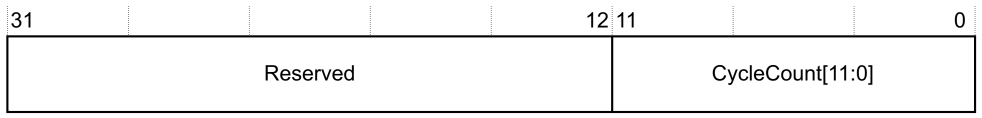


Figure 3-4 Formatter Synchronization Counter Register bit assignments

Table 3-4 shows the bit assignments.

Table 3-4 Formatter Synchronization Counter Register bit assignments

Bits	Name	Description
[31:12]	-	Reserved. RAZ on reads, SBZP on writes.
[11:0]	CycCount	12-bit counter value to indicate the number of complete frames between full synchronization packets. Because formatting is not present in the TPIU-Lite, this register is set to `0x000` at reset, and all writes are ignored.

3.5 Integration test registers

Integration test registers are provided to simplify the process of verifying the integration of the TPIU-Lite with other devices in a system. These registers enable direct control of outputs and the ability to read the value of inputs.

Note

You must only use these registers when the Integration Test Control Register bit [0] is set to 1. See *Integration Mode Control Register, 0xF00* on page 3-14.

The integration test registers are described in:

- *Integration Test Trigger In and Flush In Acknowledge Register, 0xEE4*
- *Integration Test Trigger In and Flush In Register, 0xEE8* on page 3-11
- *Integration Test ATB Data Register 0, 0xEEC* on page 3-11
- *Integration Test ATB Control Register 2, 0xEF0* on page 3-12
- *Integration Test ATB Control Register 0, 0xEF8* on page 3-13.

3.5.1 Integration Test Trigger In and Flush In Acknowledge Register, 0xEE4

The Integration Test Trigger In and Flush In Acknowledge Register, ITTRFLINACK, enables control of the **TRIGINACK** output from the TPIU-Lite. Figure 3-5 shows the bit assignments.

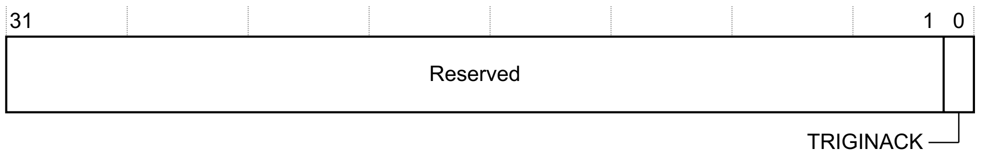


Figure 3-5 Integration Test Trigger In and Flush In Acknowledge Register bit assignments

Table 3-5 shows the bit assignments.

Table 3-5 Integration Test Trigger In and Flush In Acknowledge Register bit assignments

Bits	Name	Function
[31:1]	-	Reserved
[0]	TRIGINACK	Set the value of **TRIGINACK**

3.5.2 Integration Test Trigger In and Flush In Register, 0xEE8

The Integration Test Trigger In and Flush In Register, ITTRFLIN, contains the value of the **TRIGIN** input to the TPIU-Lite. Figure 3-6 shows the bit assignments.

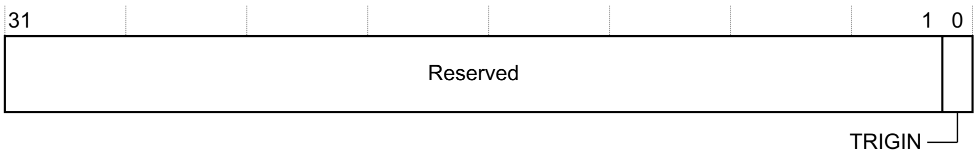


Figure 3-6 Integration Test Trigger In and Flush In Register bit assignments

Table 3-6 shows the bit assignments.

Table 3-6 Integration Test Trigger In and Flush In Register bit assignments

Bits	Type	Name	Function
[31:1]	-	-	Reserved. RAZ on reads, SBZP on writes.
[0]	RO	TRIGIN	Read the value of **TRIGIN**.

3.5.3 Integration Test ATB Data Register 0, 0xEEC

The Integration Test ATB Data Register 0, ITATBDATA0, contains the value of the **ATDATA** inputs to the TPIU-Lite. The values are only valid when **ATVALIDS** is HIGH. Figure 3-7 shows the bit assignments.

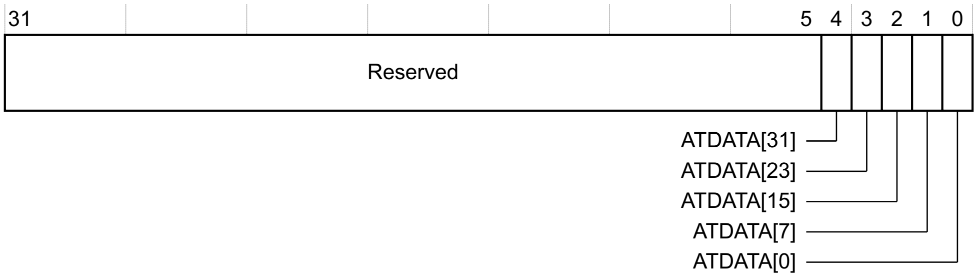


Figure 3-7 Integration Test ATB Data Register 0 bit assignments

Table 3-7 shows the bit assignments.

Table 3-7 Integration Test ATB Data Register 0 bit assignments

Bits	Name	Function
[31:5]	-	Reserved. RAZ on reads, SBZP on writes.
[4]	ATDATA[31]	Read the value of **ATDATAS[31]**.
[3]	ATDATA[23]	Read the value of **ATDATAS[23]**.
[2]	ATDATA[15]	Read the value of **ATDATAS[15]**.
[1]	ATDATA[7]	Read the value of **ATDATAS[7]**.
[0]	ATDATA[0]	Read the value of **ATDATAS[0]**.

3.5.4 Integration Test ATB Control Register 2, 0xEF0

The Integration Test ATB Control Register 2, ITATBCTR2, enables control of the **ATREADYS** and **AFVALIDS** outputs of the TPIU-Lite. Figure 3-8 shows the bit assignments.

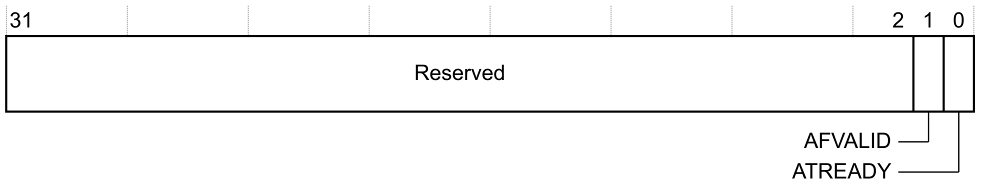


Figure 3-8 Integration Test ATB Control Register 2 bit assignments

Table 3-8 shows the bit assignments.

Table 3-8 Integration Test ATB Control Register 2 bit assignments

Bits	Name	Function
[31:2]	-	Reserved
[1]	AFVALID	Set the value of **AFVALIDS**
[0]	ATREADY	Set the value of **ATREADYS**

3.5.5 Integration Test ATB Control Register 0, 0xEF8

The Integration Test ATB Control Register 0, ITATBCTR0, 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. Figure 3-9 shows the bit assignments.

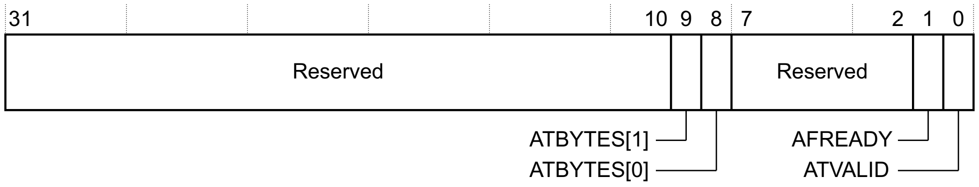


Figure 3-9 Integration Test ATB Control Register 0 bit assignments

Table 3-9 shows the bit assignments.

Table 3-9 Integration Test ATB Control Register 0 bit assignments

Bits	Name	Function
[31:10]	-	Reserved. RAZ on reads, SBZP on writes.
[9]	ATBYTES[1]	Read the value of **ATBYTESS[1]**.
[8]	ATBYTES[0]	Read the value of **ATBYTESS[0]**.
[7:2]	-	Reserved. RAZ on reads, SBZP on writes.
[1]	AFREADY	Read the value of **AFREADYS**.
[0]	ATVALID	Read the value of **ATVALIDS**.

3.6 CoreSight-defined registers

The CoreSight-defined registers are described in:

- *Integration Mode Control Register, 0xF00*
- *Claim Tag Set Register, 0xFA0* on page 3-15
- *Claim Tag Clear Register, 0xFA4* on page 3-15
- *Lock Access Register, 0xFB0* on page 3-16
- *Lock Status Register, 0xFB4* on page 3-16
- *Authentication Status Register, 0xFB8* on page 3-17
- *Device ID Register, 0xFC8* on page 3-18
- *Device Type Identifier Register, 0xFCC* on page 3-19
- *Peripheral ID Registers, 0xFD0-0xFEC* on page 3-19
- *Component ID Registers, 0xFF0-0xFFC* on page 3-24.

3.6.1 Integration Mode Control Register, 0xF00

The Integration Mode Control Register, ITCTRL, enables the component to switch from a functional mode, which is default behavior, into integration mode where the inputs and outputs of the component can be directly controlled for integration testing or topology solving. For information on topology solving, see the *CoreSight Architecture Specification*.

Figure 3-10 shows the bit assignments.

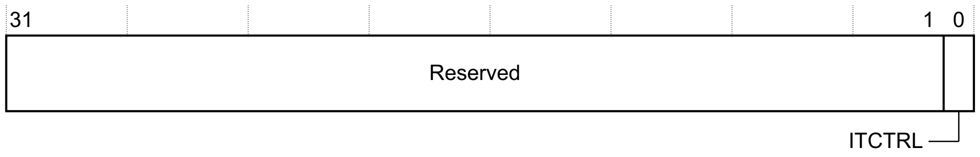


Figure 3-10 Integration Mode Control Register bit assignments

Table 3-10 shows the bit assignments.

Table 3-10 Integration Mode Control Register bit assignments

Bits	Name	Description
[31:1]	-	Reserved. RAZ on reads, SBZP on writes.
[0]	ITCTRL	Set to `0x1` to enable integration test mode.

3.6.2 Claim Tag Set Register, 0xFA0

The Claim Tag Set Register returns the number of bits that can be set on a read, and enables individual bits to be set on a write. This register is used with the Claim Tag Clear Register. Table 3-11 shows the bit assignments.

Table 3-11 Claim Tag Set Register bit assignments

Bits	Description
[31:4]	Reserved. RAZ on reads, SBZP on writes.
[3:0]	A bit-programmable register bank that reads the *Claim Tag Value* (CTV) Reads: A read of `32'h0000000F` indicates that this claim tag is four bits wide. Logic 1 indicates that this bit can be set. Logic 0 indicates that this bit is unimplemented, that is, it cannot be set. Writes: Bit-clear has no effect on the CTV. Bit-set marks the bit in the CTV.

3.6.3 Claim Tag Clear Register, 0xFA4

The Claim Tag Clear Register is used in conjunction with the Claim Tag Set Register. Table 3-12 shows the bit assignments.

Table 3-12 Claim Tag Clear Register bit assignments

Bits	Description
[31:4]	Reserved. RAZ on reads, SBZP on writes.
[3:0]	A bit-programmable register bank that sets the CTV. Reads: A read returns a value indicating the current claim value. Writes: Bit-clear has no effect on CTV. Bit-set removes the bit in the CTV. This is reset to `0x0`.

For more information about the Claim Tag Registers, see the *CoreSight Architecture Specification*.

3.6.4 Lock Access Register, 0xFB0

The Lock Access Register enables a write access to component registers. Table 3-13 shows the bit assignments.

Table 3-13 Lock Access Register bit assignments

Bits	Description
[31:0]	Write Access Code. A write of `0xC5ACCE55` enables further write access to this component. An invalid write has the effect of removing write access. This register is only present when Lock Status Register bit [0] is HIGH.

3.6.5 Lock Status Register, 0xFB4

The Lock Status Register indicates the status of the lock control mechanism. This register is used in conjunction with the Lock Access Register.

The TPIU-Lite has two views of the Lock Status Register depending on the setting of **PADDRDBG[31]**:

- When **PADDDBG[31]** is HIGH the register reads `0x0` indicating that the unit is unlocked and there is no Lock Access Register.
- If **PADDRDBG[31]** is LOW the register reads `0x3` from reset indicating that the 32-bit Lock Access Register is present and the device is currently locked.

Figure 3-11 shows the bit assignments.

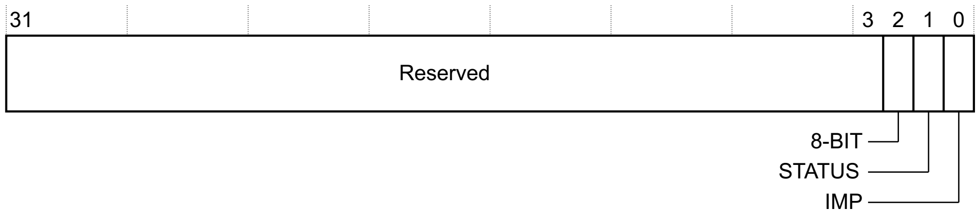


Figure 3-11 Lock Status Register bit assignments

Table 3-14 shows the bit assignments.

Table 3-14 Lock Status Register bit assignments

Bits	Name	Description
[31:3]	-	Reserved. RAZ on reads, SBZP on writes.
[2]	8-BIT	1 = Device implements an 8-bit lock register. 0 = Device implements a 32-bit lock register.
[1]	STATUS	Lock status: 1 = Device write access is locked. 0 = Device write access is granted.
[0]	IMP	Indicates that a lock control mechanism exists for this component: 1 = Lock mechanism is implemented. 0 = Lock mechanism is not implemented.

3.6.6 Authentication Status Register, 0xFB8

The Authentication Status Register reports the required security level and current status of the security enable bit pairs. Where functionality changes on a given security level, this change must be reported in this register. This register is set to `0x00` for functionality not implemented.

Figure 3-12 shows the bit assignments.

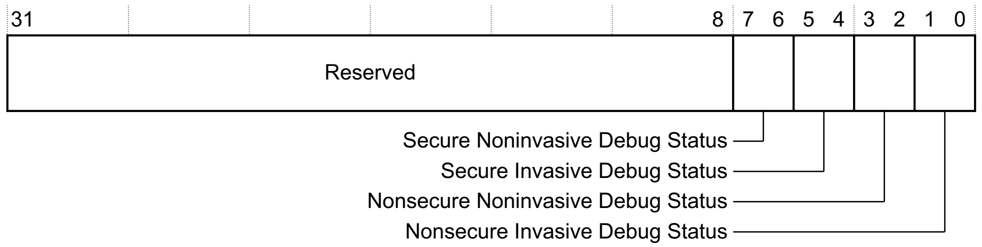

31
8
7
6
5
4
3
2
1
0
Reserved
Secure Noninvasive Debug Status
Secure Invasive Debug Status
Nonsecure Noninvasive Debug Status
Nonsecure Invasive Debug Status

Figure 3-12 Authentication Status Register bit assignments

Table 3-15 shows the bit assignments.

Table 3-15 Authentication Status Register bit assignments

Bits	Description
[31:8]	Reserved. RAZ on reads, SBZP on writes.
[7:6]	Secure Noninvasive Debug.
[5:4]	Secure Invasive Debug.
[3:2]	Nonsecure Noninvasive Debug.
[1:0]	Nonsecure Invasive Debug.

The description for each pair of bits in each debug set is shown in Table 3-16.

Table 3-16 Authentication Status Register pairs

Value	Description
2'b00	Functionality not implemented or controlled elsewhere.
2'b01	Reserved.
2'b10	Functionality disabled.
2'b11	Functionality enabled.

3.6.7 Device ID Register, 0xFC8

Table 3-17 shows the decoding of the Device ID Register.

Table 3-17 Device ID Register bit assignments

Bits	Value	Description
[31:13]	`0x00000`	Reserved
[12]	0	STP not fitted
[11]	0	Serial Wire Output (UART and NRZ) not supported
[10]	0	Serial Wire Output (Manchester) not supported
[9]	0	Trace Clock + Data is supported

Table 3-17 Device ID Register bit assignments (continued)

Bits	Value	Description
[8:6]	`0x0`	FIFO Size of 1. FIFO not required because the Trace Out port is synchronous to **ATCLK**.
[5]	0	Indicates the relationship between **ATCLK** and **TRACECLKIN**. `0x0` indicates that they are synchronous and **TRACECLKIN** is therefore not present.
[4:0]	`0x00`	Hidden Level of Input Muxing. `0x00` reports no muxing present.

3.6.8 Device Type Identifier Register, 0xFCC

The Device Type Identifier Register enables the TPIU-Lite to be identified by type.

Figure 3-13 shows the bit assignments.

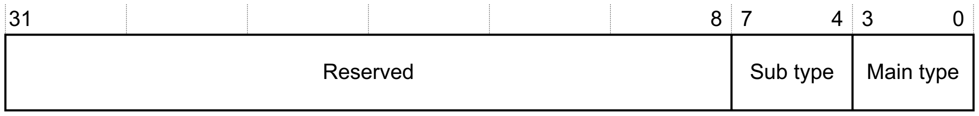


Figure 3-13 Device Type Identifier Register bit assignments

Table 3-18 shows the bit assignments.

Table 3-18 Device Type Identifier Register bit assignments

Bits	Description
[31:8]	Reserved. RAZ on reads, SBZP on writes.
[7:4]	Sub type. Set to `0x1` to indicate a TPIU.
[3:0]	Main type. Set to `0x1` to indicate a trace sink.

3.6.9 Peripheral ID Registers, 0xFD0-0xFEC

The Peripheral ID Registers indicate the following information:

Part Number (0xFE4 [3:0], 0xFEO [7:4], 0xFEO [3:0])

The three hex values represent the upper, middle, and lower value of the component device number. The TPIU-Lite has the value `0x941`.

JEP106 Identity (0xFD0 [3:0], 0xFE8 [2:0], 0xFE4 [7:4])

JEP106 continuation code (4 bits) and JEP106 identity code (7 bits). This is an ARM-designed component and takes the values `0x4` (continuation code) and `0x3B` (identity code).

Revision (0xFE8 [7:4])

The value indicates the revision of the component starting from `0x0`, corresponding to r0p0. The value increments for each part number on a revision, minor or major.

Customer Modified (0xFEC [3:0])

This component has no customer-modified RTL and has a value of `0x0`.

RevAnd (0xFEC [7:4])

This is changed from `0x0` if any alterations have to be made to the device during layout.

The Peripheral ID Registers are described in:

- *Peripheral ID4 Register, 0xFD0*
- *Peripheral ID5 Register, 0xFD4* on page 3-21
- *Peripheral ID6 Register, 0xFD8* on page 3-21
- *Peripheral ID7 Register, 0xFDC* on page 3-21
- *Peripheral ID0 Register, 0xFE0* on page 3-21
- *Peripheral ID1 Register, 0xFE4* on page 3-22
- *Peripheral ID2 Register, 0xFE8* on page 3-22
- *Peripheral ID3 Register, 0xFEC* on page 3-23.

Peripheral ID4 Register, 0xFD0

The Peripheral ID4 Register is hard-coded and the fields in the register determine the reset value.

Figure 3-14 shows the bit assignments.

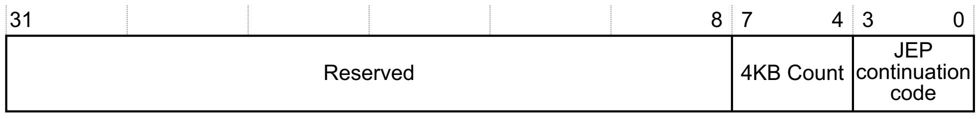


Figure 3-14 Peripheral ID4 Register bit assignments

Table 3-19 shows the bit assignments.

Table 3-19 Peripheral ID4 Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved RAZ
[7:4]	4KB count	Read as `0x0` for TPIU-Lite
[3:0]	JEP continuation code	Read as `0x4` for ARM Limited

Peripheral ID5 Register, 0xFD4

The Peripheral ID5 Register is unused and is reserved for future use.

Peripheral ID6 Register, 0xFD8

The Peripheral ID6 Register is unused and is reserved for future use.

Peripheral ID7 Register, 0xFDC

The Peripheral ID6 Register is unused and is reserved for future use.

Peripheral ID0 Register, 0xFE0

The Peripheral ID0 Register is hard-coded and the fields in the register determine the reset value.

Figure 3-15 shows the bit assignments.

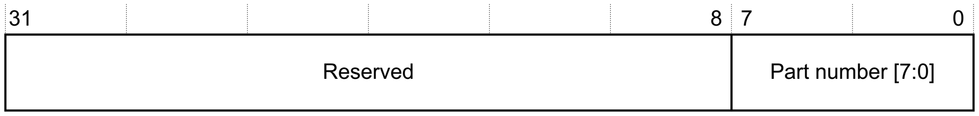


Figure 3-15 Peripheral ID0 Register bit assignments

Table 3-20 shows the bit assignments.

Table 3-20 Peripheral ID0 Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved RAZ
[7:0]	Part number[7:0]	Read as `0x41` for TPIU-Lite

Peripheral ID1 Register, 0xFE4

The Peripheral ID1 Register is hard-coded and the fields in the register determine the reset value.

Figure 3-16 shows the bit assignments.

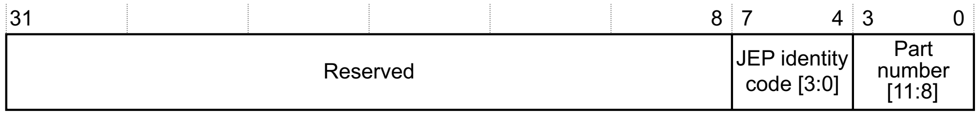


Figure 3-16 Peripheral ID1 Register bit assignments

Table 3-21 shows the bit assignments.

Table 3-21 Peripheral ID1 Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved RAZ
[7:4]	JEP identity code [3:0]	Read as `0xB` for ARM Limited
[3:0]	Part number [11:8]	Read as `0x9` for TPIU-Lite

Peripheral ID2 Register, 0xFE8

The Peripheral ID2 Register is hard-coded and the fields in the register determine the reset value.

Figure 3-17 on page 3-23 shows the bit assignments.

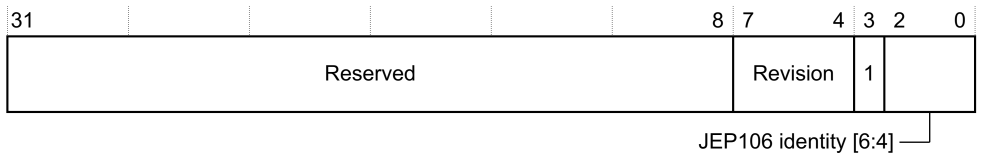


Figure 3-17 Peripheral ID2 Register bit assignments

Table 3-22 shows the bit assignments.

Table 3-22 Peripheral ID2 Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved RAZ.
[7:4]	Revision	Read as the TPIU-Lite revision, starting at `0x0` for r0p0
[3]	1	Always `0x1`, indicating a JEP106 value is used
[2:0]	JEP106 Identity [6:4]	Read as `0x3` for ARM Limited

Peripheral ID3 Register, 0xFEC

The Peripheral ID3 Register is hard-coded and the fields in the register determine the reset value.

Figure 3-18 shows the bit assignments.

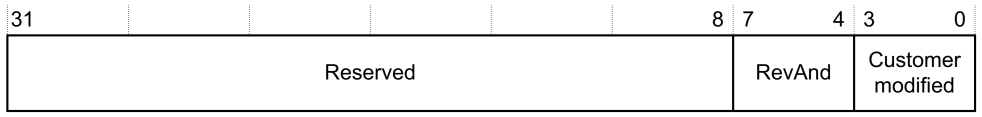


Figure 3-18 Peripheral ID3 Register bit assignments

Table 3-23 shows the bit assignments.

Table 3-23 Peripheral ID3 Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved RAZ
[7:4]	RevAnd	Read as `0x0` for TPIU-Lite
[3:0]	Customer modified	Read as `0x0` for TPIU-Lite

3.6.10 Component ID Registers, 0xFF0-0xFFC

The Component ID Registers are four 8-bit registers that span address locations `0xFF0-0xFFC`. These read-only registers can conceptually be treated as a single 32-bit register. The register is used as a standard cross-peripheral identification system.

The identification registers are described in the following sections:

- *Component ID0 Register, 0xFF0*
- *Component ID1 Register, 0xFF4*
- *Component ID2 Register, 0xFF8* on page 3-25
- *Component ID3 Register, 0xFFC* on page 3-25.

Component ID0 Register, 0xFF0

The Component ID0 register is hard-coded and the fields within the register determine the reset value.

Table 3-24 shows the bit assignments.

Table 3-24 Component ID0 Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved RAZ
[7:0]	Component ID0	These bits read back as `0x0D`

Component ID1 Register, 0xFF4

The Component ID1 register is hard-coded and the fields within the register determine the reset value.

Table 3-25 shows the bit assignments.

Table 3-25 Component ID1 Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved RAZ
[7:0]	Component ID1	Bits [7:4] read back as `0x9` Bits [3:0] read back as `0x0`

Component ID2 Register, 0xFF8

The Component ID2 register is hard-coded and the fields within the register determine the reset value.

Table 3-26 shows the bit assignments.

Table 3-26 Component ID2 Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved RAZ
[7:0]	Component ID2	These bits read back as `0x05`

Component ID3 Register, 0xFFC

The Component ID3 register is hard-coded and the fields within the register determine the reset value.

Table 3-27 shows the bit assignments.

Table 3-27 Component ID3 Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved RAZ
[7:0]	Component ID3	These bits read back as `0xB1`

Appendix A
TPIU-Lite Ports

This appendix describes the port and interface signals in the CoreSight TPIU-Lite. It contains the following sections:

A.1 ATB port

The TPIU-Lite can only be connected to a single trace source such as an ETM. There is no provision for tracing from multiple sources. Table A-1 shows the full interface.

Table A-1 ATB slave ports

Name	Type	Description
ATCLK	Input	Trace bus clock. Main clock for the internals of the TPIU.
ATCLKEN	Input	Clock enable for ATB.
ATRESETn	Input	Reset for the **ATCLK** domain, that is, ATB interface and formatter.
ATDATAS[31:0]	Input	Trace data 1-4 bytes valid with data always aligned to the LSB
ATVALIDS	Input	Valid signals in this cycle from the trace source.
ATBYTESS[1:0]	Input	Size of data, 1-4 bytes.
ATREADYS	Output	If there is valid data, that is, **ATVALID** HIGH, the data was accepted this cycle.
AFVALIDS	Output	Any data remaining in any buffers must be flushed.
AFREADYS	Input	Data flush complete, **ATFLUSH** can be deasserted.

A.2 Debug APB ports

Table A-2 shows the debug APB ports.

Table A-2 Debug APB ports

Name	Type	Description
PCLKDBG	Input	Debug APB clock.
PCLKENDBG	Input	Debug APB clock enable.
PRESETDBGn	Input	Debug APB interface reset, including programming registers.
PADDRDBG[11:2]	Input	Programming address. Only a 4K window is required.
PADDRDBG31	Input	HIGH for external accesses to bypass the Lock Access mechanism. Wire to **PADDRDBG[31]** in systems.
PSELDBG	Input	Indicates this device is currently being accessed.
PENABLEDBG	Input	Indicates second, and subsequent, cycles.
PWRITEDBG	Input	This access is a write transfer.
PWDATADBG[31:0]	Input	Write data bus.
PRDATADBG[31:0]	Output	Read data bus.
PREADYDBG	Output	Used to extend an APB transfer.

A.3 Trace Out port

Table A-3 shows the Trace Out port which is connected to ASIC pads. If fewer pins are connected to the pads than the full 32-bit **TRACEDATA** output, the total number must be reflected in the tie-off to **TPMAXDATASIZE** (see Table A-4 on page A-5).

Table A-3 Trace Out port signals

Name	Type	Description
TRACECLK	Output	Exported clock for **TRACEDATA** and **TRACECTL**. This is a divided clock from **ATCLK** and data changes in relation to both rising and falling edges.
TRACEDATA[31:0]	Output	Output data.
TRACECTL	Output	Indicates triggers and non-valid cycles on **TRACEDATA**.

A.4 Miscellaneous TPIU-Lite ports

Table A-4 shows the miscellaneous TPIU-Lite ports.

Table A-4 Miscellaneous TPIU-Lite ports

Name	Type	Description
TRIGIN	Input	From either a *Cross Trigger Interface* (CTI) or direct from a trace source, this is used to enable the trigger to affect the output trace stream.
TRIGINACK	Output	Return response to the CTI. Acknowledgement to **TRIGIN**.
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 then this takes the value `0x0F`. This input affects the Supported Port Size Register (see *Supported Port Size Register, 0x000* on page 3-5).
SE	Input	Scan enable for DFT.

Glossary

This glossary describes some of the terms used in ARM manuals. Where terms can have several meanings, the meaning presented here is intended.

Advanced High-performance Bus (AHB)

The AMBA Advanced High-performance Bus system connects embedded processors such as an ARM core to high-performance peripherals, DMA controllers, on-chip memory, and interfaces. It is a high-speed, high-bandwidth bus that supports multi-master bus management to maximize system performance.

See also Advanced Microcontroller Bus Architecture.

Advanced Microcontroller Bus Architecture (AMBA)

AMBA is the ARM open standard for multi-master on-chip buses, capable of running with multiple masters and slaves. It is an on-chip bus specification that details a strategy for the interconnection and management of functional blocks that make up a System-on-Chip (SoC). It aids in the development of embedded processors with one or more CPUs or signal processors and multiple peripherals. AMBA complements a reusable design methodology by defining a common backbone for SoC modules. AHB, APB, and AXI conform to this standard.

Advanced Peripheral Bus (APB)

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

See also Advanced High-performance Bus.

Advanced Trace Bus (ATB)

A bus used by trace devices to share CoreSight capture resources.

AHB *See* Advanced High-performance Bus.

AMBA *See* Advanced Microcontroller Bus Architecture.

APB *See* Advanced Peripheral Bus.

Application Specific Integrated Circuit (ASIC)

An integrated circuit that has been designed to perform a specific application function. It can be custom-built or mass-produced.

ATB *See* Advanced Trace Bus.

Byte An 8-bit data item.

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

Cross Trigger Interface (CTI)

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

CTI *See* Cross Trigger Interface.

CTM *See* Cross Trigger Matrix.

Embedded Cross Trigger (ECT)

The ECT is a modular component to support the interaction and synchronization of multiple triggering events with an SoC.

Embedded Trace Macrocell (ETM)

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

ETM *See* Embedded Trace Macrocell.

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

Reserved	A field in a control register or instruction format is reserved if the field is to be defined by the implementation, or produces Unpredictable results if the contents of the field are not zero. These fields are reserved for use in future extensions of the architecture or are implementation-specific. All reserved bits not used by the implementation must be written as 0 and read as 0.
SBO	*See* Should Be One.
SBZ	*See* Should Be Zero.
SBZP	*See* Should Be Zero or Preserved.
Should Be One (SBO)	Should be written as 1 (or all 1s for bit fields) by software. Writing a 0 produces Unpredictable results.
Should Be Zero (SBZ)	Should be written as 0 (or all 0s for bit fields) by software. Writing a 1 produces Unpredictable results.
Should Be Zero or Preserved (SBZP)	Should be written as 0 (or all 0s for bit fields) by software, or preserved by writing the same value back that has been previously read from the same field on the same processor.
TPA	*See* Trace Port Analyzer.
TPIU	*See* Trace Port Interface Unit.
Trace port	A port on a device, such as a processor or ASIC, used to output trace information.
Trace Port Analyzer (TPA)	A hardware device that captures trace information output on a trace port. This can be a low-cost product designed specifically for trace acquisition, or a logic analyzer.
Trace Port Interface Unit (TPIU)	The TPIU is used to drain trace data and acts as a bridge between the on-chip trace data and the data stream captured by a TPA.
Unpredictable	For reads, the data returned when reading from this location is unpredictable. It can have any value. For writes, writing to this location causes unpredictable behavior, or an unpredictable change in device configuration. Unpredictable instructions must not halt or hang the processor, or any part of the system.