Arm® CoreSight™ SDC-600 Secure Debug Channel

Revision: r0p1

Technical Reference Manual

arm

101130_0001_00_en

Arm® CoreSight™ SDC-600 Secure Debug Channel

Technical Reference Manual

Release Information

Document History

Issue	Date	Confidentiality	Change
0001-00	09 March 2018	Non-Confidential	First release for r0p1

Non-Confidential Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

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.

Unrestricted Access is an Arm internal classification.

Product Status

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

Web Address

http://www.arm.com

Contents

Arm® CoreSight™ SDC-600 Secure Debug Channel Technical Reference Manual

Preface

This preface introduces the *Arm® CoreSight™ SDC-600 Secure Debug Channel Technical Reference Manual*.

It contains the following:

About this book

This document gives reference information for the SDC-600 Secure Debug Channel.

Product revision status

The r*m*p*n* identifier indicates the revision status of the product described in this book, for example, r*1*p*2*, where:

r*m* Identifies the major revision of the product, for example, r1.

p*n* Identifies the minor revision or modification status of the product, for example, p2.

Intended audience

This book is written for system designers, system integrators, and programmers who are designing or programming a *System-on-Chip* (SoC) that uses the Arm® SDC-600 Secure Debug Channel product.

Using this book

This book is organized into the following chapters:

Chapter 1 Introduction
This chapter provides an overview of the Arm CoreSight SDC-600 Secure Debug Channel.

Chapter 2 Functional description
This chapter describes the functionality of the SDC-600.

Chapter 3 Programmers model
This chapter provides a description of the address map and registers of the SDC-600.

Appendix A Signal descriptions
This chapter describes the signals of the SDC-600.

Appendix B Revisions
This appendix describes 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 the *Arm® Glossary* for more information.

Typographic conventions

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 following figure 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.

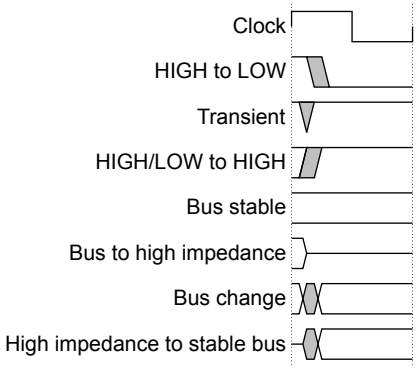

Figure 1 Key to timing diagram conventions

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.

Lowercase n

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

Additional reading

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

Arm publications
- *Arm® AMBA® APB Protocol Specification* (ARM IHI 0024C)
- *Arm® CoreSight™ Architecture Specification v3.0* (ARM IHI 0029E)
- *Arm® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2* (ARM IHI 0031D)
- *Arm® Debug Interface Architecture Specification ADIv6.0* (ARM IHI 0074A)
- *Arm® AMBA® 5 AHB Protocol Specification* (ARM IHI 0033B.b)
- *Arm® AMBA® Low Power Interface Specification* (ARM IHI 0068C)

Other publications

- *Verilog-2001 Standard (IEEE Std 1364-2001).*
- *Accellera, IP-XACT version 1685-2009.*

Feedback

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 *Arm CoreSight SDC-600 Secure Debug Channel Technical Reference Manual*.
- The number 101130_0001_00_en.
- If applicable, the page number(s) to which your comments refer.
- 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 provides an overview of the Arm CoreSight SDC-600 Secure Debug Channel.

It contains the following sections:

1.1 About

Arm CoreSight SDC-600 Secure Debug Channel provides a dedicated channel for authentication between an external debugger and a debug target as an unlocking mechanism.

The SDC-600-based architecture provides an interface through which secure debug certificates can be injected to the platform. This is done in a standard way through the *Debug Access Port* (DAP), which is normally used to debug the platform. It eliminates the need for OEM proprietary delivery mechanisms for such certificates.

SDC-600 performs the following tasks:

- Transports messages from an external debugger to a hardware or software agent on a target system through a point-to-point link.
- Establishes and maintains a link between a port on the external side, which is serviced by the debugger, and a port on the internal side, which is serviced by the target.
- Requests power and resets the target agent.

The authentication process can involve a hardware- or software-based cryptographic engine on the target. The cryptographic engine verifies the debug certificate that is deposited by the core, through the SDC-600, to the on-chip SRAM. The debugger and cryptographic engine drivers run a protocol on top of the SDC-600, which:

1. Identifies the SoC (SOC_ID).
2. Injects the appropriate debug certificate to the debug target for processing by the cryptographic engine.

The following is a high-level description of a sample authentication process:

1. The debugger wants to access the target's debug resources.
2. The debugger runs discovery and identifies the SDC-600 on COM-AP.
3. The debugger accesses the SDC-600 to start the unlocking process.
4. The SDC-600 requests the power up of the rest of its functional blocks.
5. The debugger asks for a SoC_ID to identify the target system.
6. A key is generated for the SoC_ID that is transmitted to the target.
7. The target software decides whether the debugger has the rights to access the target based on the provided key.
8. If access is granted, the target system asserts the authentication signals properly on the *Access Points* so that the connected devices can be accessed by the debugger.

1.2 Compliance

SDC-600 is compatible with the following protocol specifications and standards:

- *Arm® CoreSight™ Architecture Specification v3.0*
- *Arm® AMBA® APB Protocol Specification* (ARM IHI 0024C)
- *Arm® AMBA® 5 AHB Protocol Specification* (ARM IHI 0033B.b)
- *Arm® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2* (ARM IHI 0031D)
- *Arm® Debug Interface Architecture Specification ADIv6.0* (ARM IHI 0074A)
- *Arm® AMBA® Low Power Interface Specification* (ARM IHI 0068C)
- *IEEE 1685-2009 IP-XACT v1.5*

1.3 Configurations

SDC-600 Secure Debug Channel is designed to be integrated into three different types of systems.

These systems are:

- ADIv6-compliant systems, including implementations built using Arm CoreSight SoC-600.
- ADIv5.2-compliant systems, including implementations built using Arm CoreSight SoC-400.
- Systems that have cores with Integrated Debug.

The components and interfaces that you use for integration differ depending on your system.

1.4 Components

SDC-600 Secure Debug Channel can be integrated into various systems, and the components that are used vary depending on the system.

SDC-600 contains the following components:

- An External APBCOM/COM-AP, which is used on the debugger side and recognized as a special access port (COM-AP) by the debugger. The External APBCOM/COM-AP has three different variants depending on the debug infrastructure version.
- An Internal APBCOM, which provides a one-to-one link to the target system.
- A COM Asynchronous Bridge, which transfers data between two different clock domains. The Asynchronous Bridge has two variants:
  - An area-optimized bridge (Direct Bridge).
  - A traditional bridge (Indirect Bridge).
- The Debug Splitter, which is only required with the following supported Integrated Debug cores:
  - M0
  - M0+
  - M7
  - M23
  - M33

Related reference

2.3 Hardware components on page 2-23

1.5 Interfaces

SDC-600 has the following external interfaces:

Table 1-1 SDC-600 interfaces

Interface name	Description
AMBA4 APB	Target slave interface on all programming interfaces and register access infrastructure.
DAPBUS	Target slave interface on all programming interfaces and register access infrastructure. **Note** This interface is used instead of APB and **DP_ABORT** in ADIv5.2-compliant systems.
Cortex-M DAP-specific protocol	Can refer to a: • Target slave interface that is compatible with the debug master interface of the integrated debug of Cortex-M class processors. • Master interface that is compatible with the debug slave interface of the Cortex-M class processors in non-AHB mode.
LPI Q-Channel	Clock and Power interface.
COM Wire interface	Used for low-cost transport of bytes in a point-to-point system to enable communication from an external debugger to a hardware or software agent on a target system.
Power Request interface	Used to request power for the target side or to send a request to reset the target.
IRQ interface	Used to send an interrupt to the target system.
DP_ABORT	An abort request causes the external side to terminate an ongoing transaction and free up its APB slave interface.

Related concepts

2.4 Interfaces on page 2-29

1.6 Product design flow and documentation

Arm recommends that you perform some of the implementation stages before integrating it into your wider SoC.

The hardware processes are as follows:

Implementation
The implementer configures and synthesizes the RTL.

Integration
The integrator connects the implemented design into a SoC.

Final SoC implementation
The process of implementing the final, fully integrated SoC in silicon. Arm provides only guidance relevant to its own products for this process. If Arm provides guidance on this process for your product, then a separate document is included in the implementation bundle for that product.

This section contains the following subsection:
- *1.6.1 Documentation* on page 1-17.

1.6.1 Documentation

Each Arm document has an intended audience and is associated with specific tasks in the design flow. These documents do not reproduce Arm architecture and protocol information.

The Arm documentation set is as follows:

Technical Reference Manual
The Technical Reference Manual (TRM) describes the functionality and the effects of functional options on the behavior of the SDC-600. It is required at all stages of the design flow. The choices that are made in the design flow can mean that some behaviors that are described in the TRM are not relevant. If you are programming the SDC-600, then contact the implementer to determine:
- The build configuration of the implementation.
- What integration, if any, was performed before implementing the SDC-600.

Configuration and Integration Manual
The Configuration and Integration Manual (CIM) describes:
- A list of the design-time configuration options.
- The available build configuration options and related issues in selecting them.
- How to configure the Register Transfer Level (RTL) with the build configuration options.
- How to run test vectors.
- The processes to sign off the configured design.

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

The CIM is a confidential book that is only available to licensees.

1.7 Product revisions

This section describes the differences in functionality between product revisions of Arm.

r0p1

First release of SDC-600.

Chapter 2
Functional description

This chapter describes the functionality of the SDC-600.

It contains the following sections:

2.1 About the functions

This section describes the functional blocks in the SDC-600.

The functional blocks differ depending on the configuration.

The following figure shows the functional blocks when integrating the SDC-600 into ADIv6-compliant systems.

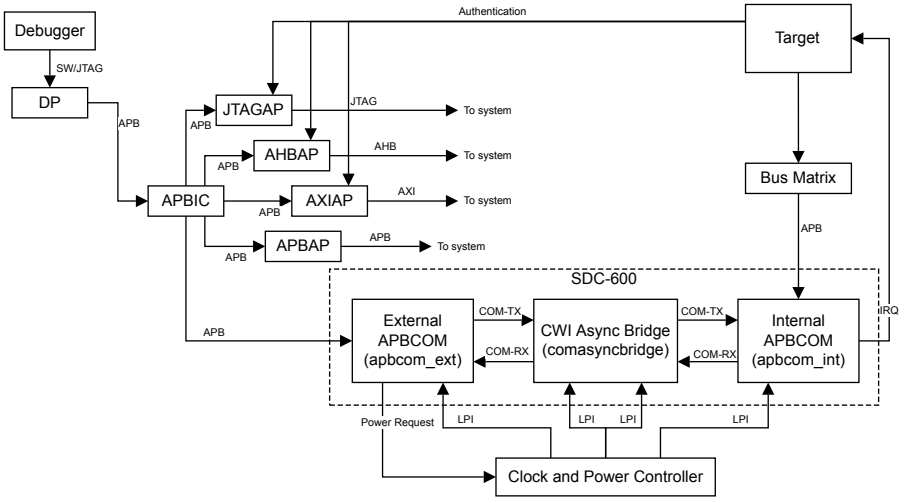

Figure 2-1 SDC-600 block diagram for ADIv6

The following figure shows the functional blocks when integrating the SDC-600 into ADIv5.2-compliant systems.

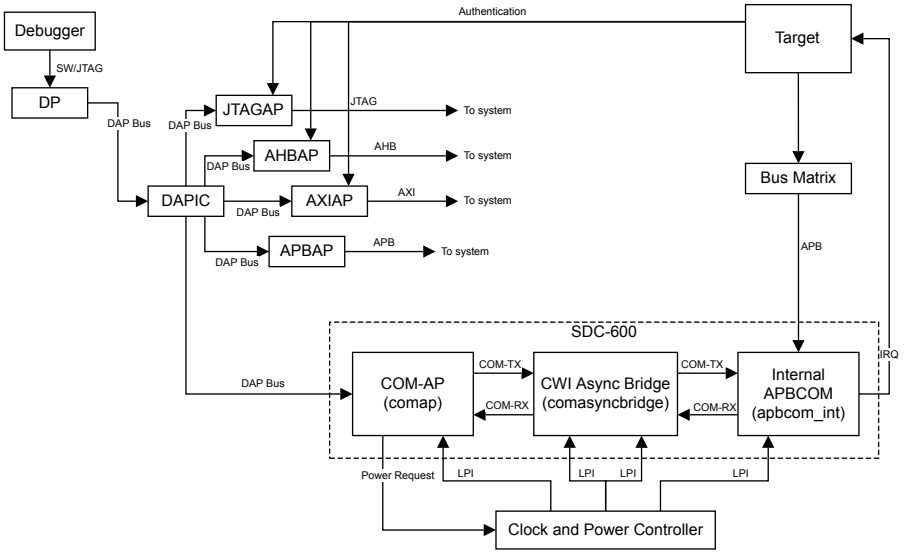

Figure 2-2 SDC-600 block diagram for ADIv5.2

The following figure shows the functional blocks when integrating the SDC-600 into systems with Integrated Debug.

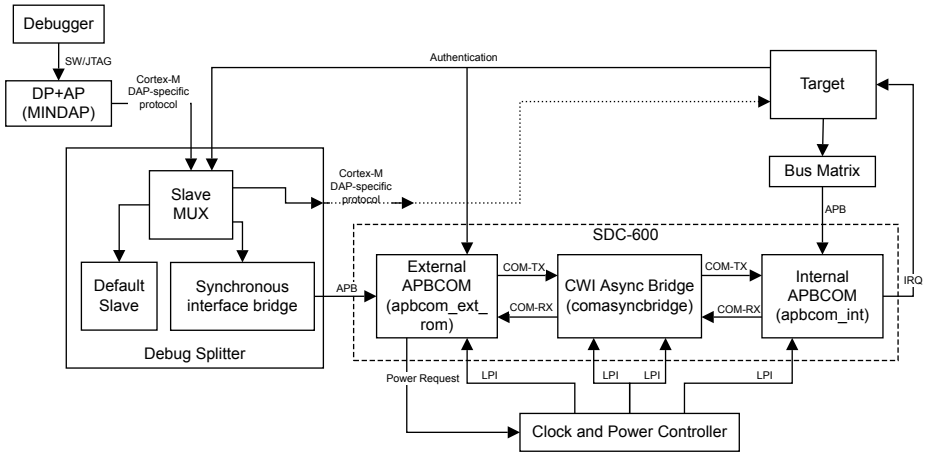

Figure 2-3 SDC-600 block diagram for Integrated Debug systems

2.2 Clocks

This section describes the clocks that are used in the SDC-600.

Table 2-1 Clocks used in SDC-600

Clock	Description
PCLK/DAPCLK	Used by the Debug Splitter and External APBCOM/COM-AP modules.
HCLK	Used by the Debug Splitter. **Note** This clock must be the same clock as the PCLK on the External APBCOM, because domain crossing is not handled inside the Debug Splitter.
PCLK	Used by the Internal APBCOM.
CLK_INT	Used by the bridge modules to bridge the clocks between domains. The clocks pass from the External APBCOM/ COM-AP to the Internal APBCOM.
CLK_EXT	

2.3 Hardware components

In a typical configuration, the SDC-600 channel contains one External APBCOM/COM-AP module and one Internal APBCOM module, with a COM asynchronous bridge in between.

The SDC-600 Secure Debug Channel contains:

- An Internal APBCOM, which is used by the target system and recognized as a peripheral device. The internal side contains the following interfaces:
  - An IRQ output, which can send an interrupt to the target system.
  - An APB4 slave.
  - Two Q-Channel interfaces: one for clock and one for power.
  - Two COM wire interfaces: one for the Rx direction and one for the Tx direction.
- An External APBCOM/COM-AP, which is used on the debugger side and recognized as a special access port (COM-AP) by the debugger. This component has three different variants depending on the debug infrastructure version. The external side contains the following interfaces:
  - A Power Request interface, which can request power for the internal side and the optional bridges and request to reset the target system.
  - A **DP_ABORT** input. An abort request causes the external side to terminate an ongoing transaction and free up its APB slave interface.
  - The Authentication interface.
  - An APB4 slave.
  - Two Q-Channel interfaces: one for clock and one for power.
  - Two COM wire interfaces: one for the Rx direction and one for the Tx direction.
- COM Asynchronous Bridges, which transfer data between two different clock domains. The COM Asynchronous Bridges have the following interfaces:
  - Two COM wire interfaces on each side: one for the Rx direction and one for the Tx direction.
  - Two clock LPIs. The Indirect Bridge has an additional power LPI.

If you implement the SDC-600 in an extendible debug system that is ADIv6 or ADIv5.2 compliant, no additional components are required.

If you implement the SDC-600 in a debug subsystem that cannot be extended by an additional *Access Port* (AP) or in a system that uses Integrated Debug, the Debug Splitter component is required. The Debug Splitter intercepts the AP downstream interface, and splits it into a downstream interface for the SDC-600 and another downstream interface for the rest of the system. The Debug Splitter contains:

- A target slave multiplexor interface, which provides one slave interface and three master interfaces.
- A low-latency synchronous interface bridge, which includes the following interfaces:
  - A slave target interface.
  - An APB4 master interface.
- A default slave interface.

The following table shows which components are required for each variant:

Component	ADIv6	ADIv5.2	Integrated Debug
External APBCOM	Required	-	-
COM-AP	-	Required	-
Integrated Debug External APBCOM	-	-	Required
Internal APBCOM	Required	Required	Required
COM Asynchronous Bridge	Optional	Optional	Optional
Debug Splitter	-	-	Required

This section contains the following subsections:

- *2.3.1 APBCOM/COM-AP* on page 2-24.

- *2.3.2 COM Asynchronous Bridge* on page 2-26.
- *2.3.3 Debug Splitter* on page 2-27.

2.3.1 APBCOM/COM-AP

APBCOM/COM-AP is the base module of SDC-600.

The APBCOM/COM-AP module provides APB access to a duplex *COM Wire Interface* (CWI) link for low-bandwidth communication. It has one Internal and three External configurations.

This component is used on the debugger side and recognized as a special access port (COM-AP) by the debugger. The COM-AP address map is configured by the various configurations to match the debug infrastructure version. It has a Power Request interface to request power for the target side or to send a request to reset the target.

The Internal APBCOM has an IRQ output.

External APBCOM

This is the external-side APBCOM component for ADIv6-compliant systems, such as Arm CoreSight SoC-600.

This component uses an APB bus. The following figure shows the port connections:

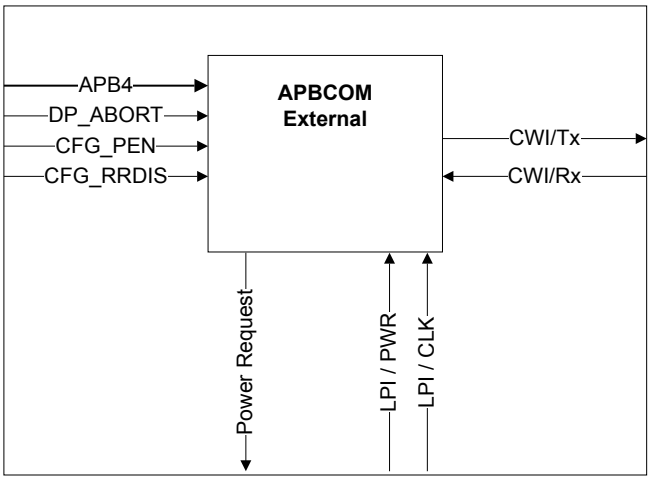

Figure 2-4 External APBCOM for ADIv6-compliant systems

COM-AP

This is the external-side COM-AP component for ADIv5.2-compliant systems, such as Arm CoreSight SoC-400.

This component uses a DAP bus. The following figure shows the port connections:

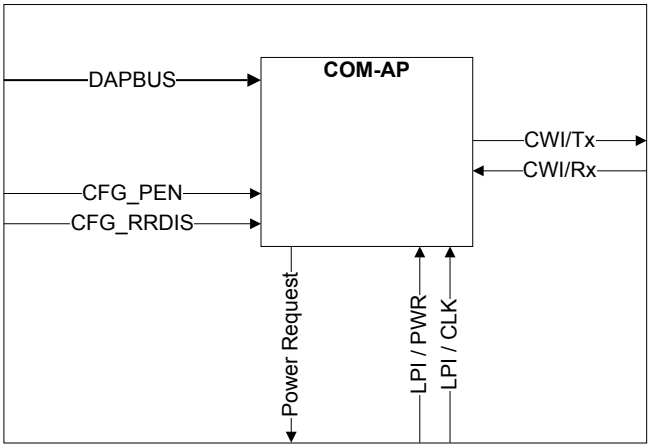

Figure 2-5 COM-AP for ADIv5.2-compliant systems

External APBCOM for systems with Integrated Debug

This is the external-side APBCOM component for systems with Integrated Debug.

This component uses an APB bus. The following figure shows the port connections:

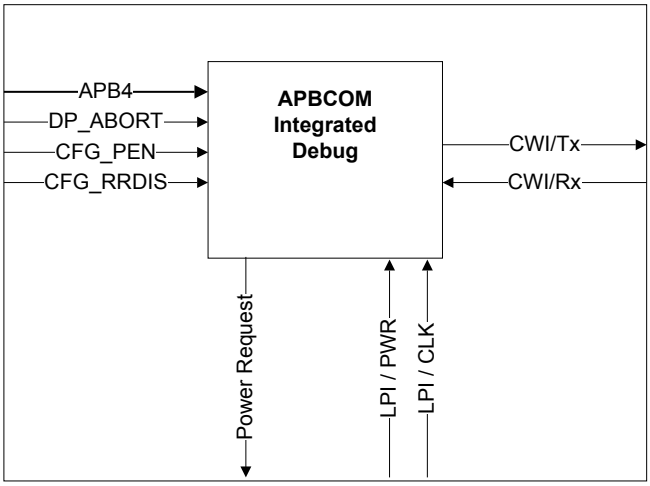

Figure 2-6 External APBCOM for systems with Integrated Debug

Internal APBCOM

The Internal APBCOM relies on the ADIv6 component for most signals.

This component uses an APB bus. The following figure shows the port connections:

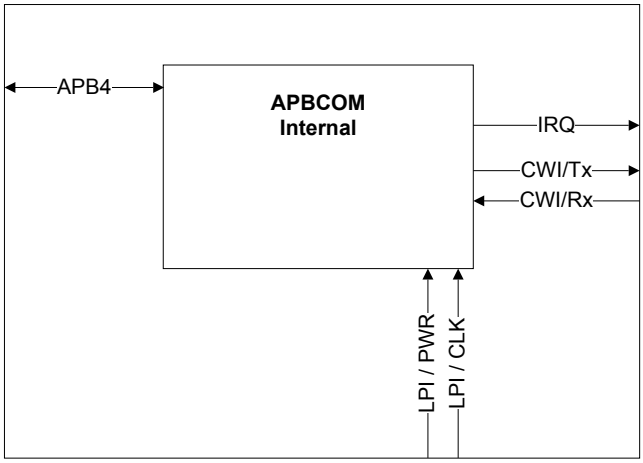

Figure 2-7 Internal APBCOM

2.3.2 COM Asynchronous Bridge

The SDC-600 COM Asynchronous Bridge module is used as a connection between the External APBCOM/COM-AP and Internal APBCOM when they are in different power or clock domains.

The bridge synchronizes the inputs coming from the clock domain of the other side. This module transfers and synchronizes data bytes between different domains using 4-way handshake signals.

The bridge is available in two versions:

Direct COM Asynchronous bridge

For this area-optimized bridge, the data is not registered on either side of the bridge. The sideband signals are not registered on the sending side, but only synchronized on the receiving side of the bridge.

It has a symmetrical structure with two CWIs on each side of the domains: one for the Rx and one for the Tx.

This bridge has two clock LPI interfaces, one for each clock domain.

——— **Note** ———

If there are multiple domain crossings, the use of indirect bridges is highly recommended.

———————————

Indirect COM Asynchronous bridge

The Indirect Bridge has an asymmetrical structure, and data is registered on both sides of the bridge. Sideband signals are all registered on the sending side and synchronized on the receiving side of the bridge. This is done to reduce constraining the CWI and floorplanning at the domain crossings.

The traditional bridge can be instantiated multiple times on a CWI channel if passing multiple domains. This bridge has two clock LPI interfaces and a power LPI interface to facilitate integration with multiple domain crossings.

The following figure shows the high-level blocks and transactions between the components when they are in asynchronous clock domains:

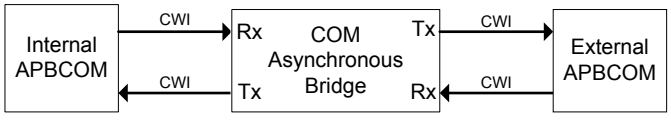

Figure 2-8 COM Asynchronous Bridge high-level block diagram

2.3.3 Debug Splitter

The Debug Splitter is a multi-unit module that is placed between the AHB master port of the integrated debug and the SLV port of the core in non-AHB-Lite compliant mode.

Note

- The Debug Splitter is not used in extendable ADIv6 or ADIv5.2 compliant debug subsystems.
- The Debug Splitter can only be used with cores with integrated debug and can only be placed between the AHB master port of the integrated debug and the SLV (debug) port of the core.

The Debug Splitter module provides access control and further branches. It provides a Cortex-M DAP-specific protocol slave interface for the Debug Access Port of the integrated debug on its debugger side. On its target system side, it has:

- A Cortex-M DAP-specific protocol master interface.
- An APB4 master interface to the SDC-600.
- An authentication input interface, through which the target system can enable the intercepted Cortex-M DAP-specific protocol master interface.

The Debug Splitter contains the following components:

- A Slave Multiplexer
- A Default Slave
- A Synchronous Interface Bridge

The following figure shows the functional blocks of the Debug Splitter.

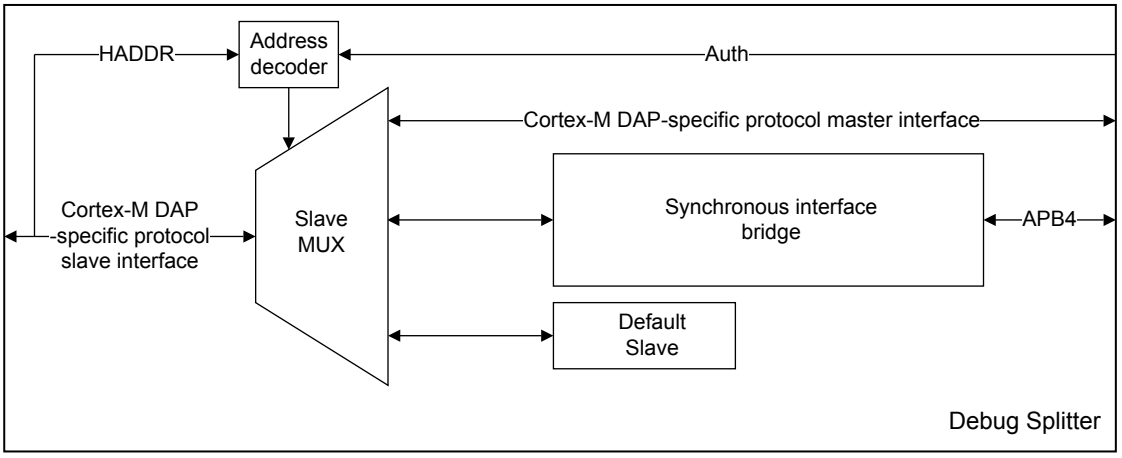

Figure 2-9 Debug Splitter block diagram

Slave Multiplexer

The Slave Multiplexer is configured to have three ports:

- One for the path towards the core.
- One for SDC-600 through the Synchronous Interface Bridge.
- One for the Default Slave.

Note

- If the debug is not enabled, transactions are forwarded towards the Default Slave instead of the core, which terminates in an error. The path is therefore blocked by the address decoder when the debug is not enabled on the Authentication interface signals.
- The multiplexer forwards all non-32-bit accesses, which are addressed to the SDC-600, to the Default Slave.

Default Slave

The Default Slave has a slave interface where only few signals are taking part in the error response generation method.

The Default Slave conforms to Default Slave behavior as described in *Arm® AMBA® 5 AHB Protocol Specification*.

Synchronous Interface Bridge

The Synchronous Interface Bridge connects a low-bandwidth APB4 periphery device to the Cortex-M DAP-specific protocol slave interface.

The Synchronous Interface Bridge has the following interfaces:

- A Slave Initiator interface to connect to the Cortex-M DAP-specific protocol slave interface.
- An APB4 Master Interface to connect to the periphery device.

Note

The clocks are common.

2.4 Interfaces

This section provides an overview of the interfaces that are used in the SDC-600 Secure Debug Channel.

The SDC-600 uses the following bus standards:

- APB4 for all programming interfaces and register access infrastructure.
- Cortex-M DAP-specific protocol for the AHB Access Port.
- Q-Channel for Low-Power Interfaces.

This section contains the following subsections:

2.4.1 Clock and power control interfaces

The External APBCOM/COM-AP and the Internal APBCOM components can be located in different clock and power domains.

APBCOM/COM-AP LPI

The SDC-600 APBCOM and COM-AP have a clock and a power LPI Q-Channel to communicate to the clock and power controller agent.

The clock LPI behaves in the following ways:

- The APB and CWI are monitored. If there is any activity, the low-power requests are denied.
- The Power Request interface is monitored. If there is a transition, the clock is preserved or requested if it is not running.
- **PWAKE_S** is asynchronously connected to **QACTIVE**. It is then possible to wake the module without a running clock.
- When there is no running clock, it can be asynchronously requested by APB, CWI, or power Q-Channel activity.
- Clock wake-up is requested when the power LPI requests the low-power state to be changed and the clocks are turned off.

The power LPI behaves in the following ways:

- Any requests are denied if there is APB or DAPBUS activity, an active Power request, or an established CWI link.
- In powerdown state, power is not requested on interface activity without additional circuitry.
- When link establishment is initiated but not yet confirmed, the module signals active on **QACTIVE**. It also accepts quiescence requests, which makes it possible to reset the internal side, if necessary.

COM Asynchronous Bridge LPI

The SDC-600 COM Asynchronous Bridges have two clock LPI Q-Channels, which communicate with the clock controller agent.

The indirect CWI asynchronous bridge module has an additional power LPI Q-Channel, which communicates with the power controller.

The clock LPI behaves in the following ways:

- If there is an outstanding transaction on the CWI, requests are denied.
- When there is no running clock, it can be asynchronously requested by CWI or power Q-Channel activity.

The power LPI behaves in the following ways:

- An established CWI link causes any requests to be denied.
- When link establishment is initiated but not yet confirmed, the module signals active on **QACTIVE**. It also accepts quiescence requests, which makes it possible to reset the internal side, if necessary.
- In powerdown state, power is not requested on interface activity without additional circuitry.

Power and Reset Request Interface

When the External APBCOM/COM-AP is on and the Internal APBCOM is off, the External side requests power for the Internal APBCOM and detects when power is granted.

The power request is maintained for the duration of communication activity, such as authentication, and removed when finished. The power request is initiated through dedicated signals according to the ADIv6 Power Request interface.

The SDC-600 External APBCOM/COM-AP can send remote reset requests to the reset controller to request the target agent to be reset. If the Internal APBCOM is located in the same reset domain as the target agent, it is also reset. The remote reset request is initiated through dedicated signals according to the ADIv6 Power Request interface.

2.4.2 Interface to the External APBCOM/COM-AP

For ADIv6-compliant systems, the APB interface connects the External APBCOM to the *Debug Port* (DP) through APB infrastructure components. The External APBCOM is preferred to be high in the APB infrastructure hierarchy.

For ADIv5.2-compliant systems, the DAPBUS interface connects the COM-AP to the Debug Port (DP) through the DAPBUS Interconnect (DAPIC).

Note

- The SDC-600 can be connected lower in the hierarchy for Arm CoreSight SoC-600 implementations. However, in this case all elements on the route to the COM-AP or External APBCOM must be accessible without any authentication. Otherwise, the communication is blocked.
- The COM-AP or External APBCOM address map is configured depending on the version of the debug infrastructure. See the Control and Status Register Map for your configuration in *Chapter 3 Programmers model* on page 3-32.

The External APBCOM also has a **DP_ABORT** signal. An abort request causes the External APBCOM to terminate the current transaction. Abort requests only affect APB write transactions to the Data Blocking Register (DBR). Other transactions, which are guaranteed to complete in a finite amount of time, ignore the abort and are allowed to complete normally.

When an abort occurs:

- The aborted transaction returns an APB error response and sets the TRINPROG bit in the Status Register.
- If SR.TRINPROG is set, any further writes to DR or DBR registers return error responses until the bit is cleared. The TRINPROG bit is cleared when the CWI transaction that was in progress when the abort request was received is completed.
- Transactions on the CWI are not affected.

2.4.3 COM Wire Interface

The connection between the External and Internal APBCOM modules is a full-duplex implementation of the *COM Wire Interface* (CWI).

To implement a full-duplex communication, two CWIs are needed between the Internal APBCOM and the External APBCOM/COM-AP of the SDC-600. This allows you to connect the Tx and Rx ports.

When the External APBCOM/COM-AP and Internal APBCOM are not in the same clock domain, synchronization is required. This is accomplished with a bridge module that converts the VALID/

READY signaling style to the REQ/ACK 4-way handshake style. In this case, one or more COM Asynchronous Bridge modules are placed between the External and Internal sides.

2.4.4 APB interface from the target to the Internal APBCOM

The interface from a target to the Internal APBCOM is APB.

The SDC-600 can generate an IRQ from the Internal APBCOM module, if enabled. FIDTXR.TXINT and FIDRXR.RXINT are set because Tx and Rx IRQ generation is implemented.

A Tx IRQ is generated if it is enabled and any of the following occurs:

- Tx FIFO is empty and ready to accept new data.
- Tx FIFO has less than a certain number of bytes remaining to process as specified by ICSR.TXFIL.

  ———— **Note** ————

  If ICSR.TXFIL is set higher than the Tx FIFO depth as defined by FIDTXR.TXFD, the Tx IRQ will be continuously generated.

  ————————————

An Rx IRQ is generated if it is enabled and any of the following occurs:

- RxEngine FIFO has at least the number of bytes ready as specified in ICSR.RXFIL.
- RxEngine FIFO is full. This occurs when the number of bytes stored reaches the value defined by FIDRXR.RXFD.

———— **Note** ————

Rx IRQ generation is enabled by default. When the link establishment is initiated by the external side, an IRQ is generated.

————————————

Chapter 3
Programmers model

This chapter provides a description of the address map and registers of the SDC-600.

It contains the following sections:

3.1 About this programmers model

This section provides general information about SDC-600 register properties.

The following information applies to Arm registers:

- The base address is not fixed, and can be different for any particular system implementation. The offset of each register from the base address is fixed.
- Do not attempt to access reserved or unused address locations. Attempting to access these locations can result in UNPREDICTABLE behavior.
- Unless otherwise stated in the accompanying text:
  - Do not modify undefined register bits.
  - Ignore undefined register bits on reads.
- Access type is described as follows:

  RW
  Read and write.

  RO
  Read only.

  WO
  Write only.

  RAZ
  Read as zero.

  WI
  Writes ignored.

The programmer's models of the External and Internal APBCOMs are the same, only the relevant capability registers have different content according to what features are required on each side. SDC-600 implements one-deep FIFOs in both Tx and Rx direction, which simplifies its programmer's model and behavior.

The control and status registers are slightly different for each External APBCOM variant:

- ADIv6 Variant : The control and status registers have an offset of `0xD00` according to ADIv6.
- ADIv5.2 Variant : The control and status registers have an offset of `0x00` according to ADIv5.2.
- Integrated Debug Variant : The control and status registers have an offset of `0xD00` according to ADIv6/ADIv5.2 and there is a standard ROM table starting at `0x000`.

3.2 Control and Status Register Map for ADIv6 systems

This variant can be used with ADIv6 debug subsystems, such as Arm CoreSight SoC-600 implementations.

The following figure shows the register map.

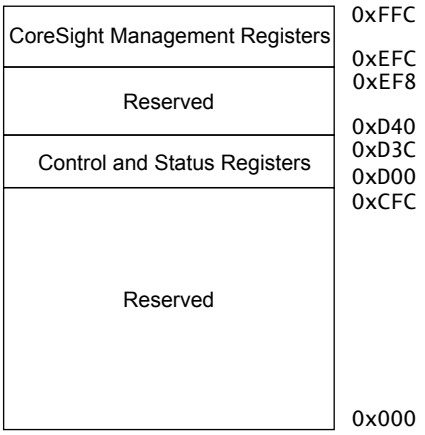

Figure 3-1 Control and Status Register map for ADIv6 debug subsystems

The following table shows the registers offset from the base memory address.

Table 3-1 Control and Status Register table for ADIv6 debug subsystems

Offset	Description
0xFFC-0xEFC	CoreSight Management Registers
0xEF8-0xD40	Reserved
0xD3C-0xD00	SDC-600 Control and Status Registers
0xCFC-0x000	Reserved

3.3 Control and Status Register Map for ADIv5.2 systems

This variant can be used with ADIv5.2 debug subsystems, such as Arm CoreSight SoC-400 implementations.

The following figure shows the register map.

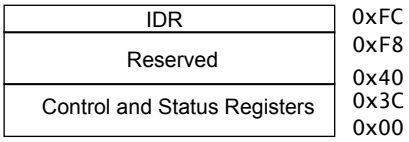

Figure 3-2 Control and Status Register map for ADIv5.2 debug subsystems

The following table shows the registers offset from the base memory address.

Table 3-2 Control and Status Register table for ADIv5.2 debug subsystems

Offset	Description
0xFC	*3.3.1 Identification Register* on page 3-35
0xF8-0x40	Reserved
0x3C-0x00	SDC-600 Control and Status Registers

This section contains the following subsection:

- *3.3.1 Identification Register* on page 3-35.

3.3.1 Identification Register

This register is used to identify the Access Port.

The IDR characteristics are:

Usage constraints
This register resets to 0x0476_2000.

Configurations
Available in the ADIv5.2 variant.

Attributes
32-bit read-only register.

The following figure shows the bit assignments.

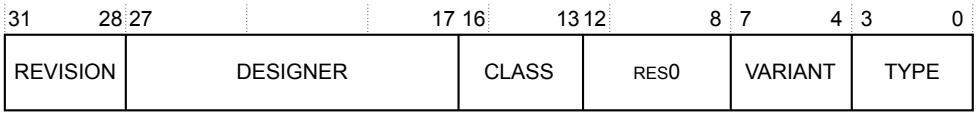

Figure 3-3 IDR bit assignments

The following table shows the bit assignments.

Table 3-3 IDR bit assignments

Bits	Name	Function
[31:28]	REVISION	0x0 Revision 0.
[27:17]	DESIGNER	0x23B Arm

Table 3-3 IDR bit assignments (continued)

Bits	Name	Function
[16:13]	CLASS	0b0001
[12:8]	-	RES0.
[7:4]	VARIANT	0b0
[3:0]	TYPE	0b0

3.4 Control and Status Register Map for systems with Integrated Debug

This variant can be used with systems with Integrated Debug.

The following figure shows the register map.

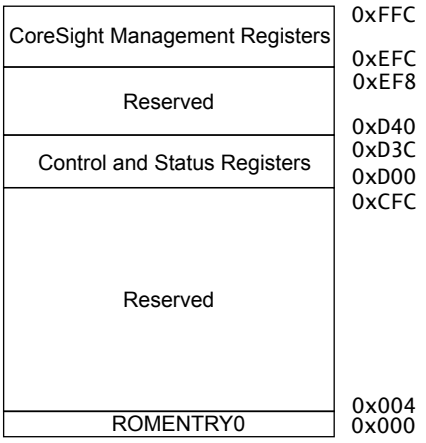

Figure 3-4 Control and Status Register map for systems with Integrated Debug

The following table shows the registers offset from the base memory address.

Table 3-4 Control and Status Register table for systems with Integrated Debug

Offset	Description
0xFFC-0xEFC	CoreSight Management Registers
0xEF8-0xD40	Reserved
0xD3C-0xD00	SDC-600 Control and Status Registers
0xCFC-0x004	Reserved
0x000	ROMENTRY0

3.5 Control and Status Registers

The following are the common control and status registers of the External APBCOM modules.

Note

Refer to the Register Map for each variant for the initial offset.

Table 3-5 Control and Status registers

Offset	Type	Size	Register	Description
0x00	RO	32	VIDR	*3.5.1 Version ID Register* on page 3-38
0x08	RO	32	FIDTXR	*3.5.2 Feature ID TxEngine Register* on page 3-39
0x0C	RO	32	FIDRXR	*3.5.3 Feature ID RxEngine Register* on page 3-41
0x10	RW	32	ICSR	*3.5.4 Interrupt Control Status Register* on page 3-42
0x20	RW	32	DR	*3.5.5 Data Register* on page 3-44
0x2C	RW	32	SR	*3.5.6 Status Register* on page 3-45
0x30	RW	32	DBR	*3.5.7 Data Blocking Register* on page 3-47
0x3C	RW	32	SR	*3.5.6 Status Register* on page 3-45

This section contains the following subsections:

- *3.5.1 Version ID Register* on page 3-38.
- *3.5.2 Feature ID TxEngine Register* on page 3-39.
- *3.5.3 Feature ID RxEngine Register* on page 3-41.
- *3.5.4 Interrupt Control Status Register* on page 3-42.
- *3.5.5 Data Register* on page 3-44.
- *3.5.6 Status Register* on page 3-45.
- *3.5.7 Data Blocking Register* on page 3-47.

3.5.1 Version ID Register

This register provides information about the architecture version of the APBCOM.

The VIDR characteristics are:

Usage constraints

There are no usage constraints.

Configurations

Available in all SDC-600 configurations.

Attributes

32-bit read-only memory-mapped register located at offset `0x00`.

This register resets to `0x00`

The following figure shows the bit assignments.

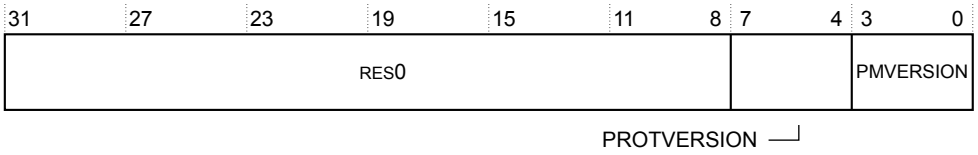

Figure 3-5 VIDR bit assignments

The following table shows the bit assignments.

Table 3-6 VIDR bit assignments

Bits	Name	Function
[31:8]	-	RES0.
[7:4]	PROTVERSION	APBCOM Protocol version. The value of this field is: `0x0` APBCOM Protocol version 0 implemented. All other values are reserved.
[3:0]	PMVERSION	APBCOM Programmers model version. The value of this field is: `0x0` APBCOM Programmers model version 0 implemented. All other values are reserved.

3.5.2 Feature ID TxEngine Register

This register provides information about the features implemented in the APBCOM TxEngine.

The FIDTXR characteristics are:

Usage constraints

There are no usage constraints.

Configurations

Available in all SDC-600 configurations.

Attributes

32-bit read-only memory-mapped register located at offset `0x08`.

The following figure shows the bit assignments.

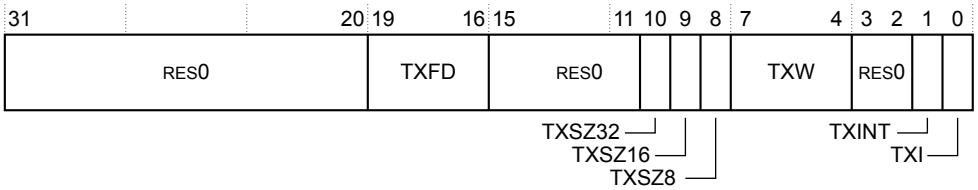

Figure 3-6 FIDTXR bit assignments

The following table shows the bit assignments.

Table 3-7 FIDTXR bit assignments

Bits	Name	Function
[31:20]	-	RES0.
[19:16]	TXFD	TxEngine FIFO depth. The defined values of this field are: `0x0-0xF` TxEngine FIFO has a capacity of at least 2^N bytes, where N is the value of this field. This field resets to `0x0`
[15:11]	-	RES0.
[10]	TXSZ32	TxEngine 32-bit write support. The defined values of this bit are: `0x0` TxEngine does not support 32-bit wide writes. `0x1` TxEngine supports 32-bit wide writes. This bit resets to `0b1`
[9]	TXSZ16	TxEngine 16-bit write support. The defined values of this bit are: `0x0` TxEngine does not support 16-bit wide writes. `0x1` TxEngine supports 16-bit wide writes. This bit resets to `0b0`
[8]	TXSZ8	TxEngine 8-bit write support. The defined values of this bit are: `0x0` TxEngine does not support 8-bit wide writes. `0x1` TxEngine supports 8-bit wide writes. This bit resets to `0b0`
[7:4]	TXW	TxEngine Width. Indicates the implemented width of the TxEngine. The defined values of this field are: `0x1` 1-byte wide TxEngine. Writes to DR or DBR will ignore bits [31:8]. `0x2` 2-byte wide TxEngine. Writes to DR or DBR will ignore bits [31:16]. `0x4` 4-byte wide TxEngine. All other values are reserved. This field resets to `0x1`
[3:2]	-	RES0.

Table 3-7 FIDTXR bit assignments (continued)

Bits	Name	Function
[1]	TXINT	Indicates whether the TxEngine generates interrupts. The defined values of this bit are: `0x0` TxEngine interrupts not implemented. `0x1` TxEngine interrupts implemented. If the TxEngine interrupts are implemented, the following are implemented: • ICSR.TXFIL • ICSR.TXFIS This bit resets to the value of TXINTEXST.
[0]	TXI	Indicates whether the TxEngine is implemented. The defined values of this bit are: `0x1` TxEngine implemented. If the TxEngine is implemented, the following are implemented: • SR.TXS • SR.TXOE • SR.TXLE • Writes to DR and DBR This bit resets to `0b1`

3.5.3 Feature ID RxEngine Register

This register provides information about the features implemented in the APBCOM RxEngine.

The FIDRXR characteristics are:

Usage constraints
There are no usage constraints.

Configurations
Available in all SDC-600 configurations.

Attributes
32-bit read-only memory-mapped register located at offset `0x0C`.

The following figure shows the bit assignments.

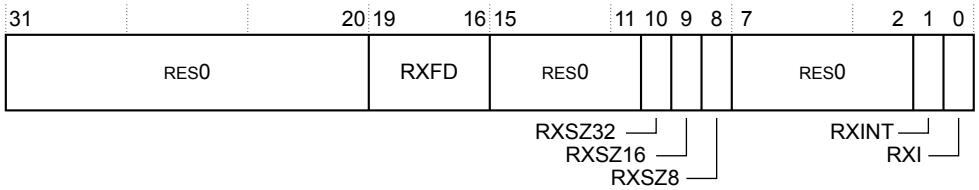

Figure 3-7 FIDRXR bit assignments

The following table shows the bit assignments.

Table 3-8 FIDRXR bit assignments

Bits	Name	Function
[31:20]	-	RES0.
[19:16]	RXFD	RxEngine FIFO depth. The defined values of this field are: `0x0-0xF` RxEngine FIFO has a capacity of at least 2^N bytes, where N is the value of this field. This field resets to `0x0`
[15:11]	-	RES0.
[10]	RXSZ32	RxEngine 32-bit write support. The defined values of this bit are: `0x0` RxEngine does not support 32-bit wide writes. `0x1` RxEngine supports 32-bit wide writes. This bit resets to `0b1`
[9]	RXSZ16	RxEngine 16-bit write support. The defined values of this bit are: `0x0` RxEngine does not support 16-bit wide writes. `0x1` RxEngine supports 16-bit wide writes. This bit resets to `0b0`
[8]	RXSZ8	RxEngine 8-bit write support. The defined values of this bit are: `0x0` RxEngine does not support 8-bit wide writes. `0x1` RxEngine supports 8-bit wide writes. This bit resets to `0b0`
[7:2]	-	RES0.
[1]	RXINT	Indicates whether the RxEngine generates interrupts. The defined values of this bit are: `0x0` RxEngine interrupts not implemented. `0x1` RxEngine interrupts implemented. If the RxEngine interrupts are implemented, the following are implemented: • ICSR.RXFIL • ICSR.RXFIS This bit resets to the value of RXINTEXST.
[0]	RXI	Indicates whether the RxEngine is implemented. The defined values of this bit are: `0x1` RxEngine implemented. If the RxEngine is implemented, the following are implemented: • SR.RXF • SR.RXLE • Reads from DR and DBR This bit resets to `0b1`

3.5.4 Interrupt Control Status Register

This register indicates the current status of the APBCOM.

The ICSR characteristics are:

Usage constraints
There are no usage constraints.

Configurations
Only present if FIDR.TXINT or FIDR.RXINT is 1. Otherwise this register is RES0.

Attributes
32-bit read/write memory-mapped register located at offset `0x10`.

The following figure shows the bit assignments.

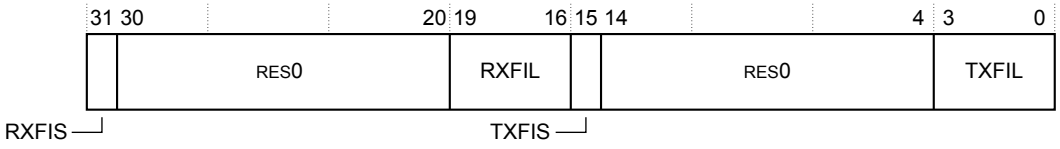

Figure 3-8 ICSR bit assignments

The following table shows the bit assignments.

Table 3-9 ICSR bit assignments

Bits	Name	Function
[31]	RXFIS	RxEngine FIFO interrupt status. The possible values of this bit are: `0x0` RxEngine FIFO interrupt has not occurred. `0x1` RxEngine FIFO interrupt has occurred. This bit is RES0 if FIDRXR.RXINT is 0. This bit is read/write-one-to-clear. This bit resets to `0x0`.
[30:20]	-	RES0.
[19:16]	RXFIL	RxEngine FIFO interrupt level select. The possible values of this field are: `0x0` RxEngine FIFO interrupts disabled. `0x1-0xF` Generate RxEngine FIFO interrupt when any of the following occur: • RxEngine FIFO has at least the specified number of bytes ready. • RxEngine FIFO is full. This occurs when the number of bytes stored reaches the value defined by FIDRXR.RXFD. This field is RES0 if FIDRXR.RXINT is 0. This field is read/write. This field resets to `0x1`.
[15]	TXFIS	TxEngine FIFO interrupt status. The possible values of this bit are: `0x0` TxEngine FIFO interrupt has not occurred. `0x1` TxEngine FIFO interrupt has occurred. This bit is RES0 if FIDTXR.TXINT is 0. This bit is read/write-one-to-clear. This bit resets to `0x0`.

Table 3-9 ICSR bit assignments (continued)

Bits	Name	Function
[14:4]	-	RES0.
[3:0]	TXFIL	TxEngine FIFO interrupt level select. The possible values of this field are: `0x0` TxEngine FIFO interrupts disabled. `0x1-0xF` Generate TxEngine FIFO interrupt when the TxEngine FIFO has less than the specified number of bytes remaining to process. This field is RES0 if FIDRXR.RXINT is 0. This field is read/write. This field resets to `0x0`.

3.5.5 Data Register

This register is used to send data via the TxEngine and receive data from the RxEngine.

The DR characteristics are:

Usage constraints
There are no usage constraints.

Configurations
Available in all SDC-600 configurations.

Attributes
32-bit read/write memory-mapped register located at offset `0x20`.

The DR and DBR operate identically, except on writes where more data is written to the TxEngine than the APBCOM can accept. In this case:

- When writing to the DR, the write access completes with an OK response and an overflow error is logged in SR.TXOE.
- When writing to the DBR, the write access stalls until there is sufficient space.

This register is:
- RW when both the TxEngine and RxEngine are implemented.
- RO when only the RxEngine is implemented.
- WO when only the TxEngine is implemented.

This register resets to `0xAFAFAFAF`.

The following figure shows the bit assignments.

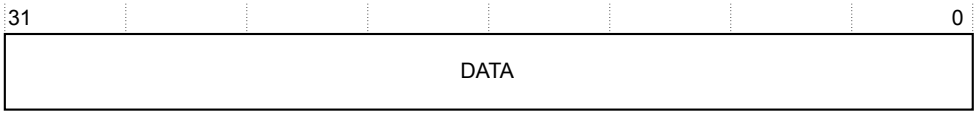

Figure 3-9 DR bit assignments

The following table shows the bit assignments.

Table 3-10 DR bit assignments

Bits	Name	Function
[31:0]	DATA	Data transfer. FIDTXR and FIDRXR indicate the supported access sizes to DBR. On writes: • Transfers bytes into the TxEngine FIFO for transmission. • Sends bytes from the least significant byte first. • If FIDTXR.TXW indicates that the TxEngine width is less than the number of bytes written, the upper unimplemented bytes are ignored by the TxEngine. The upper unimplemented bytes must be written with the NULL Flag byte value. • The TxEngine ignores any bytes that contain the NULL Flag byte value. • If there is insufficient space in the TxEngine FIFO for all of the bytes that are not NULL Flag bytes, a TxEngine Overflow error occurs and SR.TXOE is set to 1. The excess bytes are discarded by the TxEngine. • If SR.TXOE or SR.TXLE is 1, then writes are ignored. • Write accesses complete immediately. On reads: • Returns bytes from the RxEngine FIFO. • The oldest received byte is returned in the least significant byte. • If there are fewer bytes in the RxEngine FIFO than requested by the access, the upper bytes are padded by the RxEngine with the NULL Flag byte. • Read accesses complete immediately.

3.5.6 Status Register

This register indicates the current status of the APBCOM/COM-AP.

The SR characteristics are:

Usage constraints

There are no usage constraints.

Configurations

Available in all SDC-600 configurations.

Attributes

32-bit read/write memory-mapped register located at either:

- Offset `0x2C` or
- Offset `0x3C`

SR is aliased in more than one location. Accesses to any SR location access a single physical SR.

The following figure shows the bit assignments.

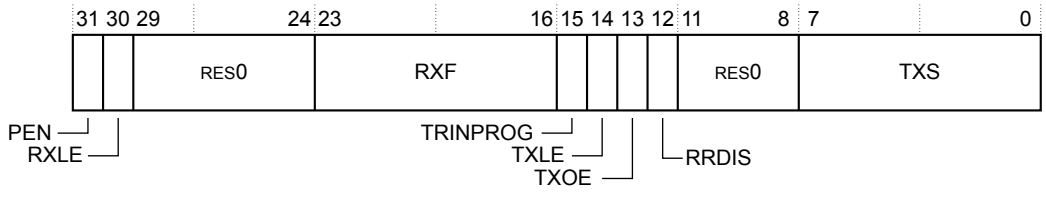

Figure 3-10 SR bit assignments

The following table shows the bit assignments.

Table 3-11 SR bit assignments

Bits	Name	Function
[31]	PEN	APBCOM/COM-AP enabled status. The defined values of this bit are: `0x0` APBCOM/COM-AP is disabled. • Writes to DR and DBR are ignored. • Reads of DR and DBR behave as if the RxEngine FIFO is empty. • Interrupt outputs are disabled. `0x1` APBCOM/COM-AP is enabled. This bit is read-only. Reading this bit returns the value of the CFG_PEN input.
[30]	RXLE	RxEngine link error detected. The possible values of this bit are: `0x0` No link error detected. `0x1` A link error has been detected in the RxEngine. This bit is RES0 if FIDRXR.RXI is 0. This bit is read/write-one-to-clear. This bit resets to `0x0`.
[29:24]	-	RES0.
[23:16]	RXF	RxEngine FIFO fill level. The possible values of this field are: `0x00` RxEngine has no data. `0x01-0xFF` RxEngine has at least the specified number of bytes available to read. This field is read-only. This field is RES0 if FIDRXR.RXI is 0. This field resets to `0x0`.
[15]	TRINPROG	Transfer in progress. The possible values of this bit are: `0x0` No transaction in progress. `0x1` An input transaction has been aborted but the internal operation of the transaction is still in progress. This bit is RES0 if the DP abort functionality is not implemented. This bit resets to `0b0`

Table 3-11 SR bit assignments (continued)

Bits	Name	Function
[14]	TXLE	TxEngine link error detected. The possible values of this bit are: `0x0` No link error detected. `0x1` A link error has been detected in the TxEngine. This bit is RES0 if FIDTXR.TXI is 0. This bit is set to 1 if: • One or more bytes written to the TxEngine are discarded because the link to the remote RxEngine is not operating. • An LERR Flag byte is inserted into the local RxEngine because the link to the remote RxEngine is not operating. This bit is read/write-one-to-clear. This bit resets to `0x0`.
[13]	TXOE	TxEngine FIFO overflow. The possible values of this bit are: `0x0` No overflow logged. `0x1` At least one byte written to TxEngine could not be accepted and has been lost. This bit is read/write-one-to-clear. This bit is RES0 if FIDTXR.TXI is 0. This bit resets to `0x0`.
[12]	RRDIS	Remote reboot requests disabled. The defined values of this bit are: `0x0` Remote Reboot requests enabled. `0x1` Remote Reboot requests disabled. When this bit is 1, the TxEngine discards any LPH2RR Flag bytes written to the TxEngine. This bit is read-only. This bit is RES0 if FIDTXR.TXI is 0. Reading this bit returns the value of the CFG_RRDIS input after synchronization.
[11:8]	-	RES0.
[7:0]	TXS	TxEngine FIFO space. The defined values of this field are: `0x0` TxEngine has no space for new data. `0x1-0xF` TxEngine has at least the specified number of bytes available. This field is RES0 if FIDTXR.TXI is 0. This field resets to `0x01` This field is read-only.

3.5.7 Data Blocking Register

This register is used to send data via the TxEngine and receive data from the RxEngine.

The DBR characteristics are:

Usage constraints

There are no usage constraints.

Configurations

Available in all SDC-600 configurations.

Attributes

32-bit read/write memory-mapped register located at offset `0x30`.

The DR and DBR operate identically, except on writes where more data is written to the TxEngine than the APBCOM can accept. In this case:

- When writing to the DR, the write access completes with an OK response and an overflow error is logged in SR.TXOE.
- When writing to the DBR, the write access stalls until there is sufficient space.

This register is:

- RW when both the TxEngine and RxEngine are implemented.
- RO when only the RxEngine is implemented.
- WO when only the TxEngine is implemented.

The following figure shows the bit assignments.

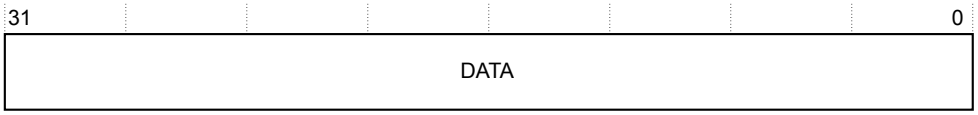

Figure 3-11 DBR bit assignments

The following table shows the bit assignments.

Table 3-12 DBR bit assignments

Bits	Name	Function
[31:0]	DATA	Data transfer. FIDTXR and FIDRXR indicate the supported access sizes to DBR. On writes: • Transfers bytes into the TxEngine FIFO for transmission. • Sends bytes from the least significant byte first. • If FIDTXR.TXW indicates that the TxEngine width is less than the number of bytes written, the upper unimplemented bytes are ignored by the TxEngine. The upper unimplemented bytes must be written with the NULL Flag byte value. • The TxEngine ignores any bytes that contain the NULL Flag byte value. • If there is insufficient space in the TxEngine FIFO for all of the bytes that are not NULL Flag bytes, the write to the DBR stalls until there is sufficient space. These writes do not complete immediately. • If SR.TXOE is 1, then writes are ignored. • If SR.TXLE is 1, then writes are ignored and any outstanding write access is terminated with an OK response. On reads: • Returns bytes from the RxEngine FIFO. • The oldest received byte is returned in the least significant byte. • If there are fewer bytes in the RxEngine FIFO than requested by the access, the upper bytes are padded by the RxEngine with the NULL Flag byte. • Read accesses complete immediately.

3.6 CoreSight™ Management Registers

Here are the CoreSight Management registers.

These registers are the same for each variant with the following exceptions:

- ITCTRL and ITSTATUS are RAZ/WI in the Internal variant.
- CLAIMSET and CLR are RAZ/WI in the Integrated Debug variant.
- AUTHSTATUS is RAZ/WI in variants other than Integrated Debug variant.

Note

For more information about these registers, see *Arm® CoreSight™ Architecture Specification v3.0*.

Table 3-13 CoreSight Management registers

Offset	Type	Size	Register	Description
0xEFC	RO	32	ITSTATUS	*3.6.1 Integration Mode Status Register* on page 3-49
0xF00	RW	32	ITCTRL	*3.6.2 Integration Mode Control Register* on page 3-50
0xFA0	RW	32	CLAIMSET	Claim Tag Set Register
0xFA4	RW	32	CLAIMCLR	Claim Tag Clear Register
0xFB8	RO	32	AUTHSTATUS	Authentication Status Register
0xFBC	RO	32	DEVARCH	Device Architecture Register
0xFC8	RO	32	DEVID	*3.6.3 Device ID Register* on page 3-51
0xFD0	RO	32	PIDR4	Peripheral ID Register
0xFE0	RO	32	PIDR0	Peripheral ID Register
0xFE4	RO	32	PIDR1	Peripheral ID Register
0xFE8	RO	32	PIDR2	Peripheral ID Register
0xFEC	RO	32	PIDR3	Peripheral ID Register
0xFF0	RO	32	CIDR0	Component ID Register
0xFF4	RO	32	CIDR1	Component ID Register
0xFF8	RO	32	CIDR2	Component ID Register
0xFFC	RO	32	CIDR3	Component ID Register

This section contains the following subsections:

- *3.6.1 Integration Mode Status Register* on page 3-49.
- *3.6.2 Integration Mode Control Register* on page 3-50.
- *3.6.3 Device ID Register* on page 3-51.

3.6.1 Integration Mode Status Register

This register is used for the Integration Test DP Abort Status.

The ITSTATUS characteristics are:

Usage constraints
There are no usage constraints.

Configurations
Available in all SDC-600 configurations.

Attributes
32-bit read/write memory-mapped register.

The following figure shows the bit assignments.

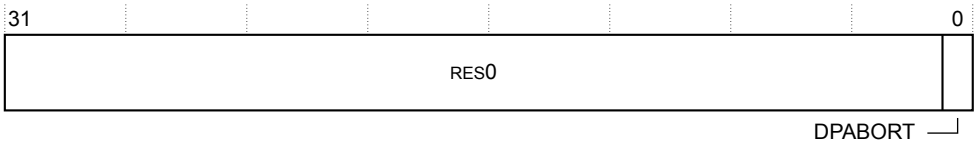

Figure 3-12 ITSTATUS bit assignments

The following table shows the bit assignments.

Table 3-14 ITSTATUS bit assignments

Bits	Name	Function
[31:1]	-	RES0.
[0]	DPABORT	In integration testing mode, when ITCTRL.IME = 1, this bit latches to 1 on rising edge of **DP_ABORT**. The possible values of this bit are: `0x0` No rising edge of **DP_ABORT** has been detected. `0x1` Rising edge has been detected on **DP_ABORT**. This bit is cleared on APB read from this register. If **DP_ABORT** rises in the same cycle when an APB read of the ITSTATUS register is received, the APB read takes priority and the register is cleared as a result. In mission mode, when ITCTRL.IME = 0, this register is RAZ/WI. This bit resets to `0x0`.

3.6.2 Integration Mode Control Register

This register is used to enable the Integration Mode.

The ITCTRL characteristics are:

Usage constraints
There are no usage constraints.

Configurations
Available in all SDC-600 configurations.

Attributes
32-bit read/write memory-mapped register.

The following figure shows the bit assignments.

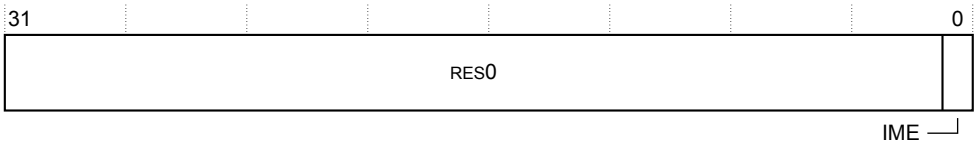

Figure 3-13 ITCTRL bit assignments

The following table shows the bit assignments.

Table 3-15 ITCTRL bit assignments

Bits	Name	Function
[31:1]	-	RES0.
[0]	IME	Integration Mode Enable. When set, the AP enters integration mode, which allows integration testing to be performed. The possible values of this bit are: `0x0` Functional mode. `0x1` Integration test mode. This bit resets to `0x0`.

3.6.3 Device ID Register

This register is used to indicate if the COM Port function is included in the ROM Table.

The DEVID characteristics are:

Usage constraints
There are no usage constraints.

Configurations
Available in all SDC-600 configurations.

Attributes
32-bit read/write memory-mapped register.

The following figure shows the bit assignments.

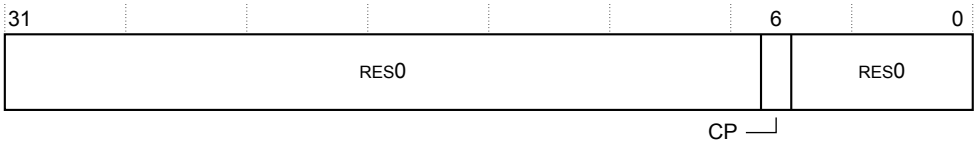

Figure 3-14 DEVID bit assignments

The following table shows the bit assignments.

Table 3-16 DEVID bit assignments

Bits	Name	Function
[31:7]	-	RES0.
[6]	CP	The Integrated Debug variant is identified as a class `0x9` ROM table. This bit means that COM port functionality is present. For all other variants, this field is 0. This bit resets to `0x1` for the Integrated Debug variant and to `0x0` for other variants.
[5:0]	-	RES0.

Appendix A
Signal descriptions

This chapter describes the signals of the SDC-600.

It contains the following sections:

A.1 APB4 External APBCOM ADIv6 signals

The following table shows the APB4 APBCOM signals for the ADIv6 variant.

Table A-1 APB4 External APBCOM ADIv6 signals

Name	Direction	Description
PCLK	Input	The main clock.
PRESET_n	Input	Active-LOW reset signal.
PWAKEUP_S	Input	Wake up request. Indicates that a new transaction is placed or about to be placed on the APB slave interface. This signal is assumed to stay asserted for the entire length of **PSEL_S**. It is guaranteed by the APB master to be glitch-free, and is used to drive **CLK_QACTIVE**. This signal is used instead of using **PSEL_S**, which could be driven from combinatorial logic, and therefore, can feed glitches into the LPI synchronizers.
CFG_RRDIS	Input	Remote reboot request disable signal from the application that drives the read only SR.RRDIS bit of the External APBCOM.
CFG_PEN	Input	Static module enable. Static input which drives the SR.PEN bit of the External APBCOM.
PADDR_S[11:0]	Input	Address bus.
PWRITE_S	Input	Indicates the direction of the data transfer. When HIGH, write is enabled. When LOW read is enabled.
PSEL_S	Input	When HIGH, the APB slave is selected.
PENABLE_S	Input	Starts the APB transfer one cycle after **PSEL**.
PWDATA_S [31:0]	Input	The data input of the APB slave when **PWRITE_S** is HIGH. The clamp value is 0.
PRDATA_S [31:0]]	Output	The selected slave drives this bus when **PWIRITE_S** is LOW. The clamp value is 0.
PREADY_S	Output	The APBCOM uses this ready signal to extend the transfer. Its inactive state is LOW. The clamp value is 0.
PSLVERR_S	Output	The APBCOM uses this error signal to indicate that an error occurred during the transfer and the transfer was aborted.

A.2 APB4 COM-AP ADIv5.2 signals

The following table shows the APB4 COM-AP signals for the ADIv5.2 variant.

Table A-2 APB4 COM-AP ADIv5.2 signals

Signal name	Direction	Description
DAPCLK	Input	The main clock.
DAPRESET_n	Input	Active-LOW reset signal.
CFG_RRDIS	Input	Remote reboot request disable signal from the application that drives the read only SR.RRDIS bit of the External APBCOM.
CFG_PEN	Input	Static module enable. Static input which drives the SR.PEN bit of the External APBCOM.
DAPCADDR_S [7:0]	Input	Address bus.
DAPWRITE_S	Input	Indicates the direction of the data transfer. When HIGH, write is enabled. When LOW read is enabled.
DAPSEL_S	Input	When HIGH, the APB slave is selected.
DAPENABLE_S	Input	Starts the APB transfer one cycle after **PSEL**.
DAPWDATA_S [31:0]	Input	The data input of the APB slave when **PWRITE_S** is HIGH. The clamp value is 0.
DAPRDATA_S [31:0]	Output	The selected slave drives this bus when **PWRITE_S** is LOW. The clamp value is 0.
DAPREADY_S	Output	The APBCOM uses this ready signal to extend the transfer. Its inactive state is LOW. The clamp value is 0.
DAPSLVERR_S	Output	The APBCOM uses this error signal to indicate that an error occurred during the transfer and the transfer was aborted.
DAPABORT_S	Input	When active-HIGH, the transfer either finishes normally in deterministic short time or is aborted.

A.3 APBCOM Q-Channel signals

The following table shows the APBCOM Q-Channel signals.

Table A-3 APBCOM Q-Channel signals

Signal name	Direction	Description
PWR_QREQn	Input	Asynchronous quiescence request signal for power.
PWR_QACCEPTn	Output	When LOW, the quiescent request is accepted for the power.
PWR_QDENY	Output	When HIGH, the quiescent request is denied for the power.
PWR_QACTIVE	Output	Active-HIGH indicates to the Q-Channel interface that the component requires power.
CLK_QREQn	Input	Asynchronous quiescence request signal for clock.
CLK_QACCEPTn	Output	When LOW, the quiescent request is accepted for the clock.
CLK_QDENY	Output	When HIGH, the quiescent request is denied for the clock.
CLK_QACTIVE	Output	Active-HIGH indicates to the Q-Channel interface that the component requires clock.

A.4 APBCOM Power Request signals

The following table shows the APBCOM Power Request signals.

Table A-4 APBCOM Power Request signals

Signal name	Direction	Description
REMPUR	Output	Remote powerup request. Indicates that the TxEngine is requesting the RxEngine to be powered up. `0x0` No remote powerup request. `0x1` Remote powerup requested.
REMPUA	Input	Remote powerup acknowledge. Indicates to the APBCOM's TxEngine that the remote RxEngine is powered up. Forms a 4-phase handshake with **REMPUR** to indicate the current status.
REMRR	Output	Remote reboot request. Indicates the APBCOM TxEngine is requesting the remote RxEngine to be rebooted. `0x0` No remote reboot request. `0x1` Remote reboot requested.
REMRA	Input	Remote reset acknowledge. Indicates to the TxEngine that the remote RxEngine is reset. Forms a 4-phase handshake with **REMRR**.

A.5 APBCOM Authentication interface signals

The following table shows Authentication interface signals that are required for the Integrated Debug variant.

Table A-5 Q-Channel signals

Signal name	Direction	Description
SPNIDEN	Input	Secure non-invasive debug enable.
SPIDEN	Input	Secure invasive debug enable.
NIDEN	Input	Non-secure non-invasive debug enable.
DBGEN	Input	Non-secure invasive debug enable.

A.6 Miscellaneous signals

The following table shows the miscellaneous signals.

Table A-6 Miscellaneous signals

Signal name	Direction	Description
DP_ABORT	Input	When an APB transfer takes more time than expected, the DP is able the abort the transfer. When active-HIGH, the transfer either finishes normally in deterministic short time or is aborted. Only exists on the External APBCOM ADIv6 and Integrated Debug variants.
IRQ	Output	Interrupt signal of the Internal APBCOM.

A.7 DEBUGSPLITTER system signals

The following table shows the Debug Splitter system signals.

Table A-7 DEBUGSPLITTER system signals

Signal name	Direction	Description
HCLK_S	Input	The bus clock times all bus transfers. All signal timings are related to the rising edge.
HRESETn_S	Input	The bus reset signal is active-LOW and resets the system and the bus.

A.8 DEBUGSPLITTER slave signals

The following table shows the Debug Splitter slave signals.

Table A-8 Debug Splitter slave signals

Name	Direction	Description
HADDR_S[31:0]	Input	Address bus with a configurable width.
HBURST_S[2:0]	Input	The burst type indicates if the transfer is a single transfer or forms part of a burst.
HMASTLOCK_S	Input	When HIGH, indicates that the current transfer is part of a locked sequence. It has the same timing as the address and control signals.
HPROT_S[3:0]	Input	The protection control signals provide additional information about a bus access and indicate how an access is handled within a system. The signals indicate if the transfer is an opcode fetch or data access. They also indicate if the transfer is a privileged mode access or a user mode access.
HPROT_S[6:4]	Input	The 3-bit extension of the **HPROT_S** signal that adds extended memory types. This signal extension is supported if the `Extended_Memory_Types` property is TRUE.
HSIZE_S[2:0]	Input	Indicates the size of the transfer, which is byte, halfword, or word.
HNONSEC_S	Input	Indicates if the current transfer is a Non-secure transfer or a Secure transfer. This signal is supported if the `Secure_Transfers` property is TRUE.
HEXCL_S	Input	Indicates that the transfer is part of an Exclusive access sequence. This signal is supported if the `Exclusive_Transfers` property is TRUE.
HMASTER_S[MASTER_WIDTH-1:0]	Input	Master identifier. Generated by a master if it has multiple Exclusive-capable threads. Modified by an interconnect to ensure that each master is uniquely identified. This signal is supported if the `Exclusive_Transfers` property is TRUE.
HTRANS_S[1:0]	Input	Indicates the transfer type of the current transfer. `IDLE`, `BUSY`, `NONSEQUENTIAL`, or `SEQUENTIAL`.
HWDATA_S[31:0]	Input	The write data bus transfers data from the master to the slaves during write operations. A minimum data bus width of 32 bits is recommended. However, the width can be extended to enable higher bandwidth operation.
HWRITE_S	Input	Indicates the transfer direction. When HIGH, this signal indicates a write transfer and when LOW a read transfer. It has the same timing as the address signals, however, it must remain constant throughout a burst transfer.
HREADY_S	Input	Transfer completion indicator.
HSEL_S	Input	Slave select signal. The **HREADYOUT_S** has to be monitored to ensure that the previous bus transfer has completed before it responds to the current transfer.
HRDATA_S[31:0]	Output	During read operations, the read data bus transfers data from the selected slave to the multiplexor. The multiplexor then transfers the data to the master. A minimum data bus width of 32 bits is recommended. However, the width can be extended to enable higher bandwidth operation.
HREADYOUT_S	Output	When HIGH, the **HREADYOUT_S** signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer.

Table A-8 Debug Splitter slave signals (continued)

Name	Direction	Description
HRESP_S	Output	The transfer response, after passing through the multiplexor, provides the master with additional information on the status of a transfer. LOW means OKAY and HIGH means ERROR.
HEXOKAY_S	Output	Exclusive Okay. Indicates the success or failure of an Exclusive Transfer. This signal is supported if the `Exclusive_Transfers` property is true.
HRUSER_S[USER_WIDTH-1:0]	Output	Read channel user signals with a configurable width.
HAUSER_S[USER_WIDTH-1:0]	Input	Address channel user signals with a configurable width.
HWUSER_S[USER_WIDTH-1:0]	Input	Write channel user signals with a configurable width.

A.9 DEBUGSPLITTER master signals

The following table shows the Debug Splitter master signals.

Table A-9 Debug Splitter master signals

Name	Direction	Description
HADDR_M[31:0]	Output	Address bus with a configurable width.
HBURST_M[2:0]	Output	The burst type indicates if the transfer is a single transfer or forms part of a burst.
HMASTLOCK_M	Output	When HIGH, indicates that the current transfer is part of a locked sequence. It has the same timing as the address and control signals.
HPROT_M[3:0]	Output	The protection control signals provide additional information about a bus access and indicate how an access is handled within a system. The signals indicate if the transfer is an opcode fetch or data access. They also indicate if the transfer is a privileged mode access or a user mode access.
HPROT_M[6:4]	Output	The 3-bit extension of the **HPROT_M** signal that adds extended memory types. This signal extension is supported if the `Extended_Memory_Types` property is TRUE.
HSIZE_M[2:0]	Output	Indicates the size of the transfer, which is byte, halfword, or word.
HNONSEC_M	Output	Indicates if the current transfer is a Non-secure transfer or a Secure transfer. This signal is supported if the `Secure_Transfers` property is TRUE.
HEXCL_M	Output	Indicates that the transfer is part of an Exclusive access sequence. This signal is supported if the `Exclusive_Transfers` property is TRUE.
HMASTER_M[MASTER_WIDTH-1:0]	Output	Master identifier. Generated by a master if it has multiple Exclusive-capable threads. Modified by an interconnect to ensure that each master is uniquely identified. This signal is supported if the `Exclusive_Transfers` property is TRUE.
HTRANS_M[1:0]	Output	Indicates the transfer type of the current transfer. `IDLE`, `BUSY`, `NONSEQUENTIAL`, or `SEQUENTIAL`.
HWDATA_M[31:0]	Output	The write data bus transfers data from the master to the slaves during write operations. A minimum data bus width of 32 bits is recommended. However, the width can be extended to enable higher bandwidth operation.
HWRITE_M	Output	Indicates the transfer direction. When HIGH, this signal indicates a write transfer and when LOW a read transfer. It has the same timing as the address signals, however, it must remain constant throughout a burst transfer.
HREADY_M	Input	Ready feedback from the multiplexor.
HSEL_M	Output	Slave select signal. The **HREADYOUT_M** has to be monitored to ensure that the previous bus transfer has completed before it responds to the current transfer.
HRDATA_M[31:0]	Input	During read operations, the read data bus transfers data from the selected slave to the multiplexor. The multiplexor then transfers the data to the master. A minimum data bus width of 32 bits is recommended. However, this can be extended to enable higher bandwidth operation.
HREADYOUT_M	Input	When HIGH, the **HREADYOUT_M** signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer.

Table A-9 Debug Splitter master signals (continued)

Name	Direction	Description
HRESP_M	Input	The transfer response, after passing through the multiplexor, provides the master with additional information on the status of a transfer. LOW means OKAY and HIGH means ERROR.
HEXOKAY_M	Input	Exclusive Okay. Indicates the success or failure of an Exclusive Transfer. This signal is supported if the `Exclusive_Transfers` property is true.
HRUSER_M[USER_WIDTH-1:0]	Input	Read channel user signals with a configurable width.
HAUSER_M[USER_WIDTH-1:0]	Output	Address channel user signals with a configurable width.
HWUSER_M[USER_WIDTH-1:0]	Output	Write channel user signals with a configurable width.

A.10 Authentication interface signals

The following table shows Authentication interface signals that are required for the Integrated Debug variant.

Table A-10 Authentication interface signals

Signal name	Direction	Description
SPNIDEN	Input	Secure non-invasive debug enable.
SPIDEN	Input	Secure invasive debug enable.
NIDEN	Input	Non-secure non-invasive debug enable.
DBGEN	Input	Non-secure invasive debug enable.

Appendix B
Revisions

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

It contains the following section:

- *B.1 Revisions* on page Appx-B-66.

B.1 Revisions

Lists the technical differences between released versions of the document.

Table B-1 Issue 0001-00

Change	Location	Affects
First release	-	-