



# DSTREAM-PT

Version 1.0

## System and Interface Design Reference Guide

**Non-Confidential**

Copyright © 2019, 2022 Arm Limited (or its affiliates).  
All rights reserved.

**Issue 03**

101714\_1.0\_03\_en



## DSTREAM-PT

### System and Interface Design Reference Guide

Copyright © 2019, 2022 Arm Limited (or its affiliates). All rights reserved.

## Release information

### Document history

Issue	Date	Confidentiality	Change
0100-00	5 April 2019	Non-Confidential	First release
0100-01	18 April 2019	Non-Confidential	Documentation update to add the Conformance Notices
0100-02	13 December 2019	Non-Confidential	Documentation update for version 1.0 release
0100-03	10 October 2022	Non-Confidential	Documentation update for version 1.0 release

## Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>.

Copyright © 2019, 2022 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349|version 21.0)

## 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.

## Feedback

Arm® welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on <https://support.developer.arm.com>

To provide feedback on the document, fill the following survey: <https://developer.arm.com/documentation-feedback-survey>.

## Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

Previous issues of this document included language that can be offensive. We have replaced this language.

To report offensive language in this document, email [terms@arm.com](mailto:terms@arm.com).

## Conformance Notices

This section contains conformance notices.

### Federal Communications Commission Notice

This device is test equipment and consequently is exempt from part 15 of the FCC Rules under section 15.103 (c).

### Class A

Important: This is a Class A device. In residential areas, this device may cause radio interference. The user should take the necessary precautions, if appropriate.

### CE/UKCA Conformity

These marks indicate that this product meets all essential health, safety and environmental requirements. The CE mark indicates conformity within EU member states and the UKCA mark indicates conformity within the UK.

The Declarations of Conformity are available on request.



The *Waste Electrical and Electronic Equipment* (WEEE) marking, that is, the crossed out wheeie-bin figure, indicates that this product must not be disposed of with general waste within the European Union. To prevent possible harm to the environment from uncontrolled waste disposal, the user is required to recycle the product responsibly to promote reuse of material resources. To comply with EU law, you must dispose of the product in one of the following ways:

- Return it to the distributor where it was purchased. The distributor is required to arrange free collection when requested.
- Recycle it using local WEEE recycling facilities. These facilities are now very common and might provide free collection.
- If purchased directly from Arm, Arm provides free collection. Please e-mail [weee@arm.com](mailto:weee@arm.com) for instructions.
- End-of-Life products can be disposed of safely using an *Approved Authorized Treatment Facility* (AATF). To support safe disposal, Arm has partnered with B2B Compliance. B2B can be contacted at the following weblink: <https://b2bcompliance.org.uk>

During the lifetime of the product, you are advised to:

- Inspect the product regularly to ensure that it is in good working order.
- Ensure that the product is free from dust and debris that might cause damage.
- Clean the product with an air duster when necessary.
- Power down the system when not in use.
- Observe ESD precautions when handling the product.

The product can radiate Radio Frequency Interference (RFI) or Electromagnetic Interference (EMI) and might cause harmful interference to radio communications. There is no guarantee that interference cannot occur in a particular installation. If you suspect that this equipment is causing interference to other equipment, you are encouraged to try to correct the interference by one or more of the following measures:

- Ensure attached cables do not lie across any sensitive equipment.
- Increase the distance between the product and the receiver.
- Connect the equipment to an outlet on a circuit different from that to which the product is connected.
- Consult Arm for help.

The product can be sensitive to Radio Frequency Interference (RFI) or Electromagnetic Interference (EMI) which might cause incorrect operation of the product:

- Avoid using the product near sources of EMI.
- Never use the product in *Safety-Critical-Systems* (SCS), or *Life-Critical-Systems* (LCS).



Arm recommends that, wherever possible, shielded interface cables be used.

---

# Contents

**List of Figures..... 8**

**List of Tables..... 10**

<b>1. Introduction.....</b>	<b>11</b>
1.1 Conventions.....	11
1.2 Useful resources.....	12
1.3 Other information.....	12
<b>2. Debug and trace interface.....</b>	<b>14</b>
2.1 JTAG signals.....	14
2.2 Return Clock signal.....	20
2.3 Reset signals.....	21
2.4 Run-Control signals.....	22
2.5 Serial Wire Debug signals.....	23
2.6 Trace signals.....	25
2.7 Target Voltage Reference signals.....	27
2.8 I/O diagrams for DSTREAM-PT signals.....	28
2.9 Typical SWD circuit.....	30
2.10 Typical JTAG circuit.....	31
<b>3. Target interface connectors.....</b>	<b>32</b>
3.1 Target connector selection guide.....	32
3.2 Arm JTAG 20 connector.....	33
3.3 CoreSight 10 connector.....	34
3.4 CoreSight 20 connector.....	35
3.5 TI JTAG 14 connector.....	37
3.6 Mictor 38 connector.....	38
3.7 Dual Mictor connectors.....	40
3.8 MIPI 34 connector.....	42
3.9 MIPI 60 connector.....	44
3.10 Auxiliary connector.....	46
3.11 User I/O connector.....	46
<b>4. Target board design.....</b>	<b>48</b>
4.1 Overview of high-speed design.....	48
4.2 JTAG port buffering.....	51
4.3 Series termination.....	54
4.4 Parallel trace modeling.....	55
4.5 Target design checklist.....	56

# List of Figures

Figure 2-1: Simple JTAG connection.....	15
Figure 2-2: Chained JTAG connection.....	16
Figure 2-3: JTAG timing diagram.....	17
Figure 2-4: Basic JTAG port synchronizer.....	18
Figure 2-5: Timing diagram for the Basic JTAG synchronizer.....	18
Figure 2-6: JTAG port synchronizer for single rising-edge D-type ASIC design rules.....	19
Figure 2-7: Timing diagram for the D-type JTAG synchronizer.....	19
Figure 2-8: Example reset circuit.....	22
Figure 2-9: SWD timing diagrams.....	24
Figure 2-10: TRACECLK timing diagram.....	26
Figure 2-11: Target interface logic levels.....	28
Figure 2-12: Input/Output signals.....	28
Figure 2-13: TCK signal.....	29
Figure 2-14: Reset signals.....	29
Figure 2-15: Trace signals.....	29
Figure 2-16: VTREF signals.....	30
Figure 2-17: Typical SWD circuit.....	30
Figure 2-18: Typical JTAG circuit.....	31
Figure 3-1: Arm JTAG 20 connector pinout.....	34
Figure 3-2: CoreSight 10 connector pinout.....	35
Figure 3-3: CoreSight 20 connector pinout.....	36
Figure 3-4: TI JTAG 14 connector pinout.....	38
Figure 3-5: Mictor 38 connector pinout.....	39



Figure 3-6: Dual Mictor connector pinout.....	41
Figure 3-7: MIPI 34 connector pinout.....	43
Figure 3-8: MIPI 60 connector pinout.....	45
Figure 3-9: User I/O connector pinout.....	47
Figure 4-1: Point-to-point signal.....	48
Figure 4-2: Stub length.....	49
Figure 4-3: Long stub causing false edges.....	49
Figure 4-4: Improved route with shorter stub.....	50
Figure 4-5: JTAG connection without buffers.....	51
Figure 4-6: JTAG connection with TDO buffer.....	52
Figure 4-7: Daisy-chained JTAG connection without buffers.....	52
Figure 4-8: Daisy-chained JTAG connection with TCK buffers.....	53
Figure 4-9: Fully buffered JTAG connection.....	53

# List of Tables

Table 2-1: JTAG timing Characteristics.....	17
Table 2-2: SWD timing requirements.....	24
Table 2-3: TRACECLK characteristics.....	26
Table 3-1: Connector attributes.....	32
Table 3-2: Arm JTAG 20 pinout table.....	34
Table 3-3: Arm CoreSight 10 pinout table.....	35
Table 3-4: Arm CoreSight 20 pinout table (DSTREAMCS20=0).....	36
Table 3-5: Arm CoreSight 20 pinout table (DSTREAMCS20=1).....	37
Table 3-6: TI JTAG 14 pinout table.....	38
Table 3-7: Mictor 38 pinout table.....	39
Table 3-8: Mictor B pinout table.....	42
Table 3-9: MIPI 34 pinout table.....	43
Table 3-10: MIPI 60 pinout table.....	45
Table 3-11: User I/O pinout table.....	47
Table 4-1: Typical series terminating resistor values.....	54
Table 4-2: Target design checklist.....	56

# 1. Introduction

DSTREAM-PT System and Interface Design Reference Guide describes the interfaces of the DSTREAM-PT debug and trace units, with details about designing Arm architecture-based devices and PCBs. This document is written for those using DSTREAM-PT or those designing PCBs.

## 1.1 Conventions

The following subsections describe conventions used in Arm documents.




### 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: [developer.arm.com/glossary](https://developer.arm.com/glossary).

### Typographic conventions

Arm documentation uses typographical conventions to convey specific meaning.

Convention	Use
<i>italic</i>	Citations.
<b>bold</b>	Interface elements, such as menu names.  Terms in descriptive lists, where appropriate.
monospace	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
monospace <u>underline</u>	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments.  For example:  <pre>MRC p15, 0, &lt;Rd&gt;, &lt;CRn&gt;, &lt;CRm&gt;, &lt;Opcode_2&gt;</pre>
<b>SMALL CAPITALS</b>	Terms that have specific technical meanings as defined in the <i>Arm® Glossary</i> . For example, <b>IMPLEMENTATION DEFINED</b> , <b>IMPLEMENTATION SPECIFIC</b> , <b>UNKNOWN</b> , and <b>UNPREDICTABLE</b> .
 Caution	Recommendations. Not following these recommendations might lead to system failure or damage.
 Warning	Requirements for the system. Not following these requirements might result in system failure or damage.
 Danger	Requirements for the system. Not following these requirements will result in system failure or damage.

Convention	Use
 Note	An important piece of information that needs your attention.
 Tip	A useful tip that might make it easier, better or faster to perform a task.
 Remember	A reminder of something important that relates to the information you are reading.

## 1.2 Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Access to Arm documents depends on their confidentiality:

- Non-Confidential documents are available at [developer.arm.com/documentation](https://developer.arm.com/documentation). Each document link in the following tables goes to the online version of the document.
- Confidential documents are available to licensees only through the product package.

Arm product resources	Document ID	Confidentiality
<a href="#">Arm Development Studio Getting Started Guide</a>	101469	Non-Confidential
<a href="#">Arm Development Studio Heterogeneous system debug with Arm Development Studio</a>	102021	Non-Confidential
<a href="#">Arm Development Studio User Guide</a>	101470	Non-Confidential
<a href="#">Arm DSTREAM-PT Getting Started Guide</a>	101713	Non-Confidential
<a href="#">Arm DSTREAM-ST Getting Started Guide</a>	100892	Non-Confidential
<a href="#">Arm DSTREAM-ST System and Interface Design Reference Guide</a>	100893	Non-Confidential

Non-Arm resources	Documentation	Organization
JTAG Specification	<a href="#">IEEE</a>	Institute of Electrical and Electronics Engineers

## 1.3 Other information

See the Arm website for other relevant information.

- [Arm® Developer](#).
- [Arm® Documentation](#).

- [Technical Support](#).
- [Arm® Glossary](#).

## 2. Debug and trace interface

The Arm debug and trace interface enables powerful software debug and optimization on an Arm® processor-based target system. It is based on the IEEE 1149.1 (JTAG) interface coupled with various additional signals. This chapter introduces these signals and describes their use within the interface.



- Unless otherwise specified, all pull-up/pull-down resistors that are discussed in this chapter must be between 1K and 100K (10K is recommended).
- Unless otherwise specified, any signals beginning with a lowercase 'n' are, by default, active-low.

### 2.1 JTAG signals

Most Arm®-based devices are physically equipped with several pins that are dedicated to debug and test purposes. Four of these pins make up the IEEE 1149.1 interface, also known as the JTAG interface. This interface is often used for boundary-scan testing during the manufacture of printed circuit boards. The interface also provides a useful way to access one or more cores and other components in a device, while running its application software.

#### Test Data In (TDI)

The **TDI** signal is an input to the target device which provides a stream of serial data from the debug unit.

The **TDI** signal must be pulled **HIGH** on the target to keep the signal inactive when no debug unit is connected.

#### Test Mode Select (TMS)

The **TMS** signal is an input to the target device which controls its JTAG state-machine.

The **TMS** signal must be pulled **HIGH** on the target to keep the signal inactive when no debug unit is connected.

#### Test Clock (TCK)

The **TCK** signal is an input to the target device which synchronizes its JTAG state-machine. On each rising edge of the **TCK** signal, the target samples the **TDI**, and **TMS** signals.

Consider **TCK** as a strobe signal, rather than a clock signal, because it is typically non-continuous and only becomes active during debug communications.

**TCK** can be pulled **HIGH** on the target, however, to maintain full compatibility with other JTAG equipment, Arm recommends you pull **TCK LOW**.

## Test Data Out (TDO)

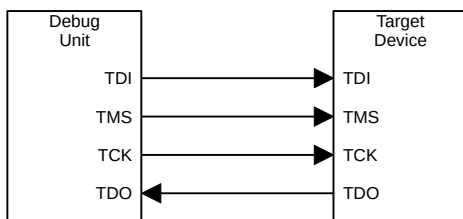
The **TDO** signal is an output from the target device which returns a stream of serial data to the debug unit.

**TDO** can be left floating on the target, however, to maintain full compatibility with other JTAG equipment, Arm recommends you pull **TDO** HIGH.

## Basic JTAG connection

In the simplest form (omitting pull-up and pull-down resistors), a connection between the debug unit and the target device looks like:

**Figure 2-1: Simple JTAG connection**



The naming convention of the **TDI** (Test Data In) and **TDO** (Test Data Out) signals is always with respect to the target device.

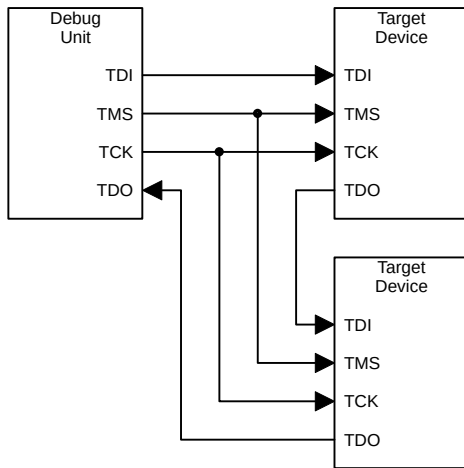


When multiple devices are used in a scan-chain, the **TCK** and **TMS** signals must be branched to each device. Good digital design practice must be used to ensure that these branches do not reduce the signal integrity of the signals causing false edges to be received by the devices.

For more information, see [JTAG port buffering](#).

The flexible design of the JTAG interface enables you to connect multiple devices to a single debug unit:

**Figure 2-2: Chained JTAG connection**



A group of JTAG devices that are linked or *daisy-chained* together is often known as the JTAG chain or scan-chain.

### JTAG timing characteristics

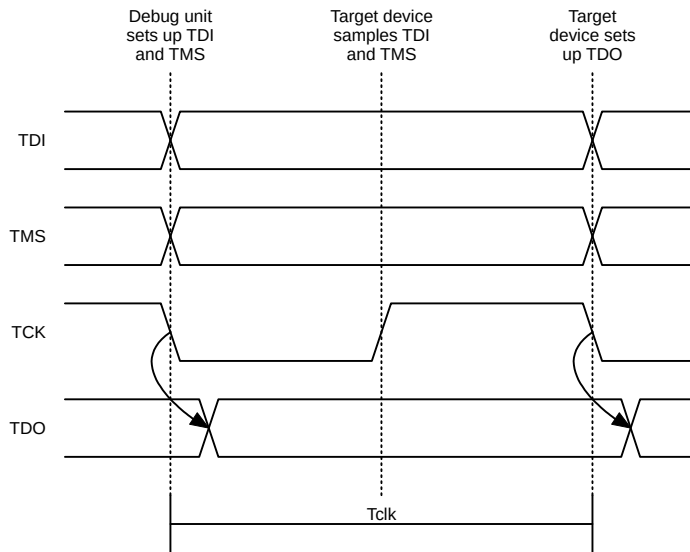
The JTAG timing characteristics of DSTREAM-PT systems conform to the requirements of the IEEE 1149.1 (JTAG) specification.

**TDI** and **TMS** are set up by the DSTREAM-PT system on the falling edge of **TCK**. These signals are then sampled by the target device on the rising edge of **TCK**. The target device must set up its **TDO** signal when it detects the falling edge of **TCK** which, in turn, will be sampled by the DSTREAM-PT system on the next rising edge of **TCK**.

These timings are considered correct at the debug connector of the target board.

Basic JTAG timing:



**Figure 2-3: JTAG timing diagram**

Since all signals are set up on the falling edge of **TCK** and sampled on the rising edge, the effective setup and hold times for the target device and DSTREAM-PT system are approximately  $T_{clk}/2$ .

Issues with signal timing can usually be resolved by decreasing the **TCK** frequency. Decreasing the **TCK** frequency increases the setup and hold times.

**TDO** is always slightly delayed, relative to the other signals, because it takes a finite amount of time for the target device to detect the falling edge of **TCK** and then set up **TDO**. This slight delay, and the round-trip delay of the debug cable, are compensated for by the DSTREAM-PT system.



There are no separate timing requirements for the adaptive clocking mode. In adaptive clocking mode, the debug unit samples **TDO** on the rising edge of **RTCK** instead of **TCK**, so **TDO** timing is relative to **RTCK**.

**Table 2-1: JTAG timing Characteristics**

Parameter	Min	Max	Description
$F_{clk}$	10Hz	180MHz	<b>TCK</b> frequency
$T_{clk}$	5.556ns	100ms	<b>TCK</b> period
$T_{ds}$	49%	51%	<b>TCK</b> Duty Cycle

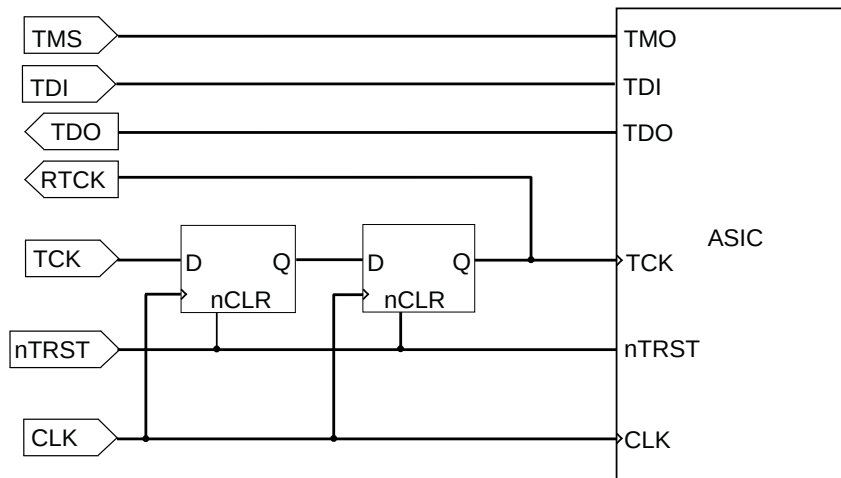
For further details on the JTAG interface, a full specification is available from: [www.ieee.org](http://www.ieee.org).

## Synchronization

As debug data is transferred to and from the target device, it must pass between two clock domains (**TCK** and the internal system clock of the target device). To achieve synchronized data transfer without suffering any meta-stability issues, a synchronizer circuit must be used within the target device.

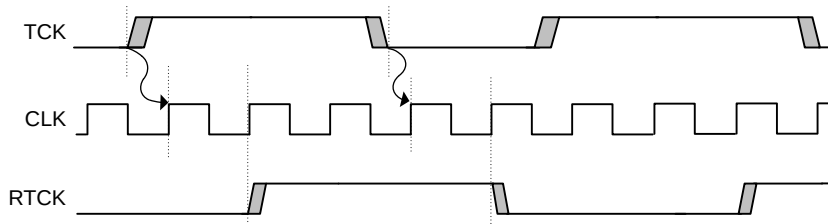
The following figure shows a circuit for a basic JTAG port synchronizer.

**Figure 2-4: Basic JTAG port synchronizer**



The following figure shows a partial timing diagram for the basic JTAG synchronizer. To reduce the delay, and because the second flip-flop only provides better immunity to metastability problems, clock the flip-flops from opposite edges of the system clock.

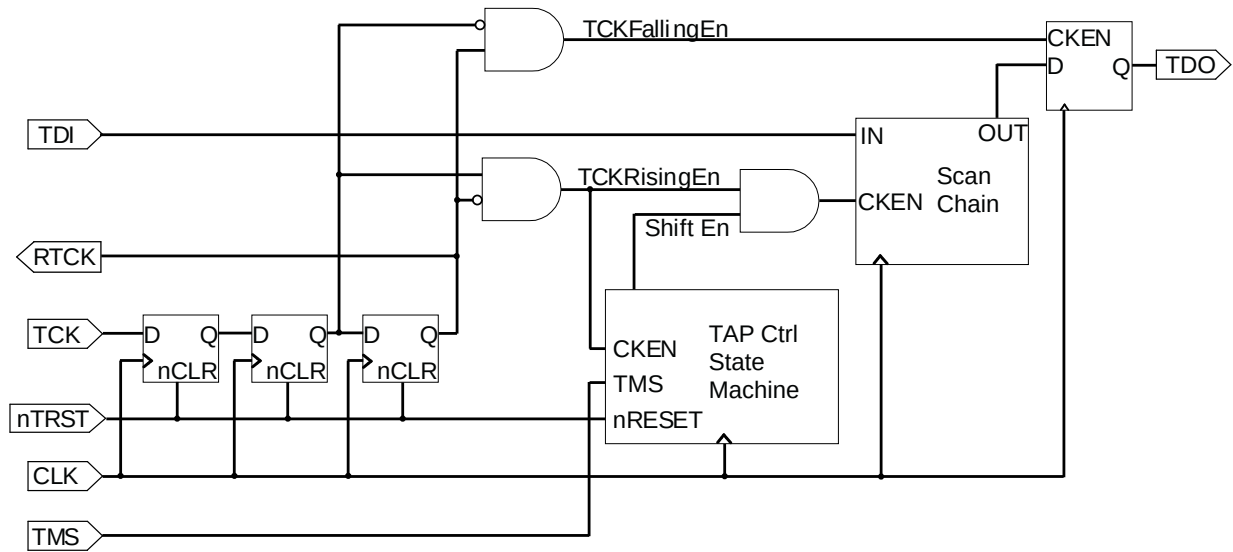
**Figure 2-5: Timing diagram for the Basic JTAG synchronizer**



ASIC design rules often impose a restriction that all flip-flops in a design must be clocked by one edge of a single clock. To interface the clocking restriction to a JTAG port that is asynchronous to the system, you must convert the JTAG **TCK** events into clock enables for this single clock. You must also ensure that the JTAG port cannot overrun this synchronization delay.

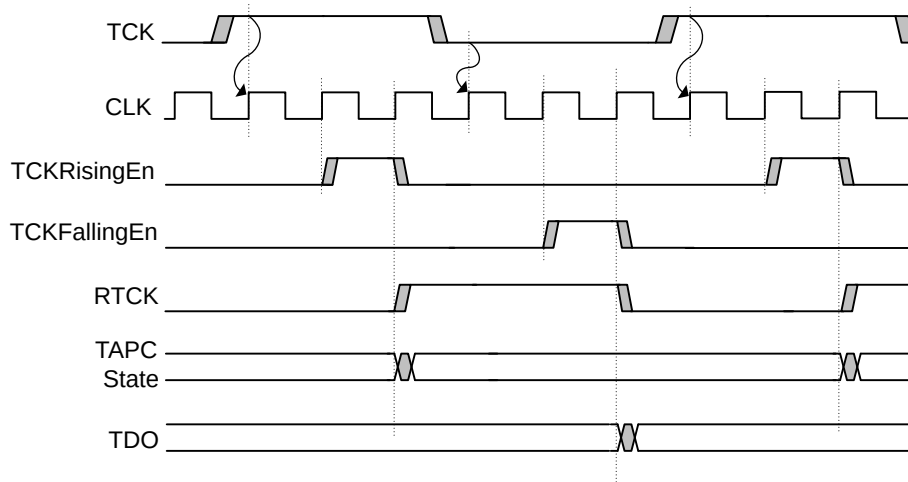
One possible implementation of this circuit, is:

**Figure 2-6: JTAG port synchronizer for single rising-edge D-type ASIC design rules**



The following figure shows a corresponding partial timing diagram, and how **TCKFallingEn** and **TCKRisingEn** are each active for exactly one period of **CLK**. It also shows how these enable signals gate the **RTCK** and **TDO** signals so that they only change state at the edges of **TCK**.

**Figure 2-7: Timing diagram for the D-type JTAG synchronizer**



## 2.2 Return Clock signal

Occasionally, a target device requires the JTAG interface to be externally synchronized to a clock within the device due to it being slow, non-continuous, or variable. The adaptive clocking feature uses the Return Clock signal (**RTCK**) to address this requirement.



**RTCK** should never be directly linked to **TCK** on the target board. If it is directly linked, it is likely to cause false clock-edges to be received by the **TCK** input of the target device.

The **RTCK** signal is an output from the target device which is typically fed from the last flip-flop in the synchronization chain.

If used, the **RTCK** signal must be pulled **LOW** on the target.

### Adaptive clocking



- If you use the adaptive clocking feature, then the transmission delays, gate delays, and synchronization requirements might result in a lower clock frequency, compared to using fixed clocking. Adaptive clocking mode is not recommended unless the target design requires it.
- If you use adaptive clocking, the debug unit cannot detect the clock speed, and therefore cannot scale its internal timeouts. If the target clock frequency is too low, a JTAG timeout might occur, leaving the JTAG interface in an unknown state. To recover the connection, you must reset the debug unit. To disable JTAG timeouts, use the configuration settings in Arm® Development Studio. For more information, see [Debug Hardware configuration](#) in the Arm Development Studio User Guide.

When adaptive clocking is enabled, the debug unit issues a **TCK** signal and waits for the **RTCK** signal to return before sampling **TDO**. The debug unit does not progress to the next **TCK** transition until **RTCK** is received, allowing the target device to control the flow of the JTAG interface, as required.



Adaptive clocking can be enabled using the configuration settings in Arm Development Studio. For more information, see [Debug Hardware configuration](#) in the Arm Development Studio User Guide.

## 2.3 Reset signals

Arm debug units have the ability to control two reset signals on the target: **nSRST** and **nTRST**.

### System Reset (nSRST)

The system reset signal, sometimes known as **nRESET** or **HRESET**, is an input to the target which performs a warm boot of the core (or cores) and other devices in the target system. It is often asserted by one or more of these conditions:

- Power On Reset (POR)
- Manual push-button reset
- Remote reset from a debug unit
- Watchdog reset

When no debug unit is connected, unintended resets can occur. To avoid unintended resets, the **nSRST** signal must be pulled to its inactive logic level on the target.

By default, the **nSRST** signal has a logic level of active **LOW**. To avoid unintended resets, pull the **nSRST** signal **HIGH**.

The polarity of the **nSRST** signal is configurable in Arm® Development Studio.

### TAP Reset (nTRST)

The TAP reset signal initializes the Test Access Port, debug logic, and boundary scan cells in the target device.

When no debug unit is connected, unintended resets can occur. To avoid unintended resets, the **nTRST** signal must be pulled to its inactive logic level on the target.

By default, the **nTRST** signal has a logic level of active **LOW**. To avoid unintended resets, pull the **nTRST** signal **HIGH**.

The polarity of the **nTRST** signal is configurable in Arm Development Studio.



Arm strongly recommends that the **nSRST** and **nTRST** signals are separately available on the JTAG connector. If the **nSRST** and **nTRST** signals are linked together, resetting the system also resets the TAP controller, which means:

- Depending on your target, it might not be possible to debug a system from reset because any breakpoints previously set might be lost.
- You might need to restart the debug session because the JTAG interface might not recover when the TAP controller state is changed.

It is expected that the assertion of the **nSRST** line by the DSTREAM-PT system will cause a warm reset of the target system. If the **nSRST** line triggers a full, Power On Reset (POR), then the debug connection might be lost.

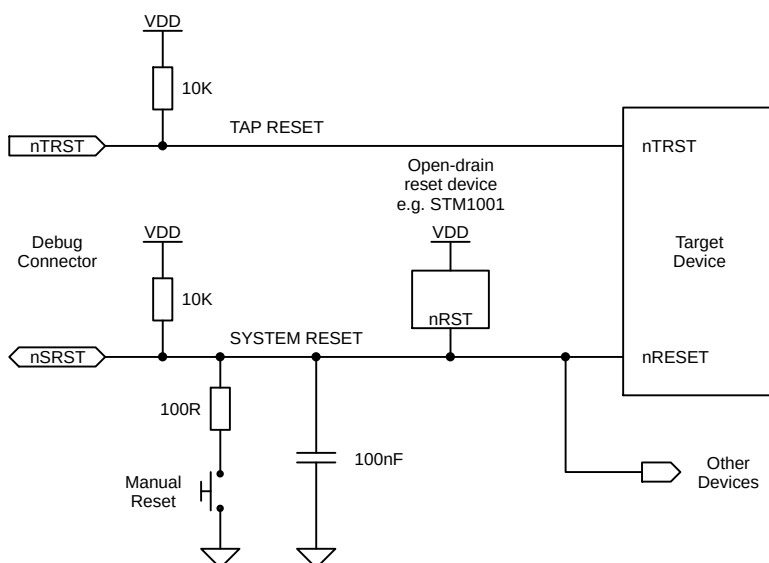
With regards to the reset signals output from the DSTREAM-PT system, the strong pull-up/pull-down resistance is approximately 33Ω, and the weak pull-up/pull-down resistance is approximately 4.7kΩ.

Because it is possible to switch the polarity and drive strength of **nTRST** and **nSRST**, target systems with various different reset configurations are supported.

### Example reset circuit

A typical reset circuit which would be present on the target board, is:

**Figure 2-8: Example reset circuit**



The push-button, 100R resistor, and 100nF capacitor shown here are an example of how a manual reset button can be interfaced with the **nSRST** signal. This is optional and would typically be used on development boards.

The reset device that is shown here would keep the target device, and any other system devices, in their reset state until the power rail has reached a minimum valid voltage. If the target device has a separate Power On Reset (POR) input, any voltage monitoring devices would typically connect to that instead. If the target device is equipped with internal voltage monitoring circuitry, external monitoring devices can be omitted.

## 2.4 Run-Control signals

The run-control signals are now deprecated within Arm® Development Studio, however, low-speed control might still be possible through ConfigItems and the command-line utilities.

### Debug Request (DBGREQ)

The Debug Request (**DBGREQ**) pin stops the target processor and puts it into its debug state.

Arm recommends that this signal is no longer used. It can be left open on the target board.



If the signal is used, it must be pulled `LOW` on the target.

---

### Debug Acknowledge (DBGACK)

The Debug Acknowledge (**DBGACK**) pin notifies the debug unit that a debug request has been received and that the target processor is now in its debug state.

Arm recommends that this signal is no longer used. It can be left open on the target board.



If the signal is used, it must be pulled `LOW` on the target.

---

## 2.5 Serial Wire Debug signals

Serial Wire Debug (SWD) is commonly used on reduced pin-count target devices. SWD only requires two pins, instead of the four pins used by JTAG.

### Serial Wire Data I/O (SWDIO)



**SWDIO** signal is bidirectional and the functionality is shared with a unidirectional JTAG **TMS** signal. Ensure that there are no buffers on the target which would prevent bidirectional communication.

---

During debugging, the Serial Wire Data I/O (**SWDIO**) signal is bidirectional. It can send and receive serial data from the target.

The **SWDIO** signal must be pulled `HIGH` on the target to keep the signal inactive when no debug unit is connected.

### Serial Wire Clock (SWCLK)

During debugging, the Serial Wire Clock (**SWCLK**) signal is an input to the target which clocks data into, and out of, the target device.

The **SWCLK** signal must be pulled `LOW` on the target to keep the signal inactive when no debug unit is connected.

## Serial Wire Output (SWO)

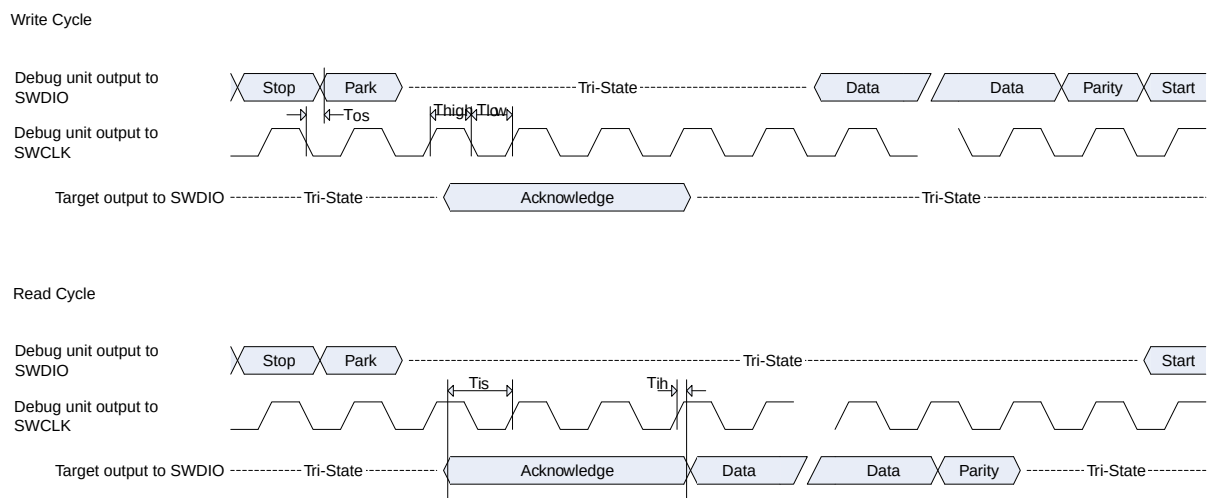
The Serial Wire Output (**SWO**) signal is an output from the target which is often used alongside the **SWD** signals to provide low-bandwidth trace.

The **SWO** signal must be pulled **HIGH** on the target to keep the signal inactive when no debug unit is connected.

## SWD timing requirements

The diagrams that are shown in the following figure separate the **SWDIO** line to show when it is driven by either the debug unit or target device:

**Figure 2-9: SWD timing diagrams**



The debug unit:

- Writes data to **SWDIO** on the falling edge of **SWCLK**.
- Reads data from **SWDIO** on the rising edge of **SWCLK**.

The target:

- Writes data to **SWDIO** on the rising edge of **SWCLK**.
- Reads data from **SWDIO** on the rising edge of **SWCLK**.

The following table shows the timing requirements for SWD:

**Table 2-2: SWD timing requirements**

Parameter	Min	Max	Description
$T_{[high]}$	4ns	50ms	<b>SWCLK</b> <sub>HIGH</sub> period.
$T_{[low]}$	4ns	50ms	<b>SWCLK</b> <sub>LOW</sub> period.
$T_{[os]}$	-1ns	1ns	<b>SWDIO</b> output skew to falling edge <b>SWCLK</b> .



Parameter	Min	Max	Description
$T_{[is]}$	4ns	-	Input setup time that is required between <b>SWDIO</b> and rising edge <b>SWCLK</b> .
$T_{[ih]}$	1ns	-	Input hold time that is required between <b>SWDIO</b> and rising edge <b>SWCLK</b> .

## 2.6 Trace signals

Some target devices can output high-bandwidth parallel trace data while the target application is running. Capturing this data and decoding it in Arm® Development Studio allows you to examine the sequence of instructions, and changes in data, around a given point or *trigger*.

For CoreSight™-compliant systems, DSTREAM-ST supports parallel trace capture of up to 4-bit wide continuous-mode Trace Port Interface Unit (TPIU) formatted trace, at up to 600Mbps per trace signal.

The Parallel Trace probe, in the DSTREAM-PT system, extends this functionality by supporting up to 32-bit wide TPIU trace.



Whilst the DSTREAM-PT system supports up to 32-bit wide trace, not all intermediate trace widths are supported. The DSTREAM-PT system supports the following trace widths: 1 - 16, 18, 20, 22, 24, 26, 28, 30 and 32 bits wide.

The trace signals supported by DSTREAM-PT are:

### TRACEDATA[0-31]

The Trace Data signals are outputs from the target and can be used to collect 1-bit to 32-bit trace data.

### TRACECLK

The Trace Clock signal is an output from the target which is used to clock the parallel trace data into the debug unit.

The trace clock signal does not need to be phase-shifted from the data signals. By default, the debug unit incorporates the appropriate delay to provide the necessary setup and hold timings for aligned **TRACEDATA** and **TRACECLK** signals.

The DSTREAM-ST only supports DDR clocking mode. Parallel trace data is captured on both the rising and falling edges of the trace clock signal.



Although the debug unit can compensate for large amounts of skew between the trace signals, to avoid the extra calibration step during configuration, Arm recommends matching the lengths of the signals within a 10mm window.

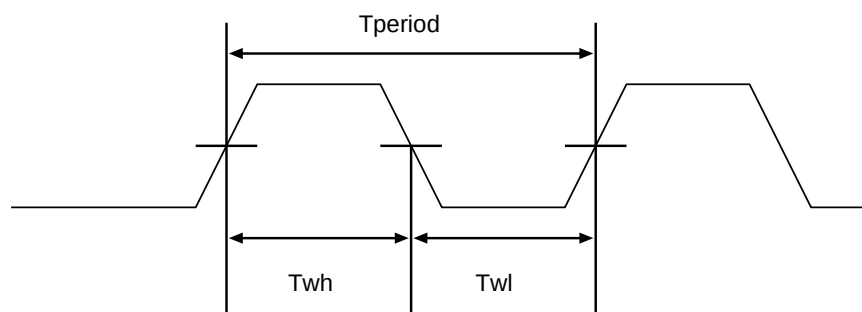
No pull-up or pull-down resistors are required for the trace signals.

To improve signal integrity, it is good practice to provide impedance matching resistors on the **TRACEDATA** and **TRACECLK** outputs close to the target device. The value of these resistors, added to the impedance of the driver, must be approximately equal to 50Ω.

To achieve the maximum data rate, Arm recommends using the short 20-way 0.05" pitch ribbon cable.

The following figure and table describe the timing for **TRACECLK**:

**Figure 2-10: TRACECLK timing diagram**



**Table 2-3: TRACECLK characteristics**

Parameter	Min	Max	Description
Tperiod (min)	1.667ns	125ms	Clock period
Twh (min)	833ps	62.5ms	High pulse width
Tlw (min)	833ps	62.5ms	Low pulse width

### Switching thresholds

The debug unit detects the target reference voltage and automatically adjusts its switching thresholds to 50% of this voltage. For example, on a 3.3V target system, the switching thresholds are set to 1.65V.

### Leakage current

If you connect an unpowered DSTREAM-ST unit to a powered target, on any of the debug or trace signals, there is a maximum leakage current into the DSTREAM-ST unit of ±10μA.

## 2.7 Target Voltage Reference signals

The Target Voltage Reference (**VTREF**) signals are used by DSTREAM-ST to determine the correct logic levels of all inputs and outputs of the debug and trace interface.

To work with debug and trace interfaces on differing voltage rails, the DSTREAM-ST unit supports separate debug and trace voltage domains.

### VTREF

When using either the CoreSight™ 20 or Arm JTAG 20 connector, only one voltage domain is supported. The voltage domain is determined using the **VTREF** signal.

### DEBUG\_VTREF

When using the Mictor adapter, or optional MIPI-34 or MIPI-60 adapters, the voltage domain of the debug signals is determined using the **DEBUG\_VTREF** signal.

### TRACE\_VTREF

When using the MIPI-60 cable, Mictor adapter, or optional MIPI-34 or MIPI-60 adapters, the voltage domain of the trace signals is determined using the **TRACE\_VTREF** signal.



Note

If only the **TRACE\_VTREF** signal is connected on a Mictor, MIPI-34, or MIPI-60 connector of a target, the DSTREAM-ST unit uses that signal to determine the logic levels of both the debug and trace signals.

Arm recommends connecting **VTREF** signals directly to one or more appropriate power rails on the target board. If a series resistor is used for short-circuit protection, the value used must be less than 100Ω.

**VTREF** signals that are received by the DSTREAM-PT system are loaded with a resistance of approximately 10K to ground. The signals are filtered, limited, and buffered to provide the required VDD (Voh) and reference voltages (Vi(th)) for the I/O stages of the debug unit.

The DSTREAM-PT system supports debug and trace logic levels between 1.2V and 3.3V.

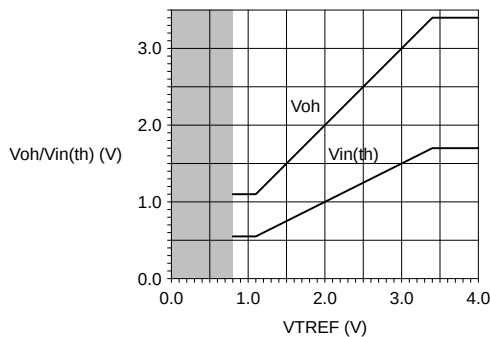


Note

- To be recognized by the DSTREAM-PT system as a valid target reference voltage, **VTREF** signals must be above 800mV. **VTREF** signals above 800mV illuminate the **VTREF** LED on the Parallel Trace probe.
- Logic levels outside the 1.2V to 3.3V window might work, but are not guaranteed to work because the DSTREAM-ST unit and Parallel Trace probe internally limit the **VTREF** signal to a minimum of approximately 1.1V, and a maximum of approximately 3.4V.

The relationships of Voh and Vi(th) to **VTREF** are:

**Figure 2-11: Target interface logic levels**



The input and output characteristics of the DSTREAM-PT system are compatible with logic levels from TTL-compatible, or CMOS logic in target systems.

## 2.8 I/O diagrams for DSTREAM-PT signals

The following diagrams and descriptions illustrate a simplified view of how each signal type is connected within the debug unit.

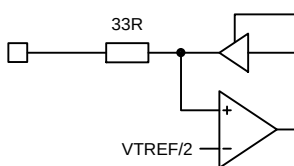


All comparator inputs have an indeterminate band of 100mV above, and below, **VTREF/2**. Signals that are output from the target system, when passing through this voltage region, must be monotonic.

### Input/Output signals

Standard input/output signals (**TDI**, **TMS**, **TDO**, **RTCK**, **SWDIO**, **DBGREQ**, **DBGACK**) use **LVC MOS** output buffers and comparator inputs with a series 33Ω resistor.

**Figure 2-12: Input/Output signals**

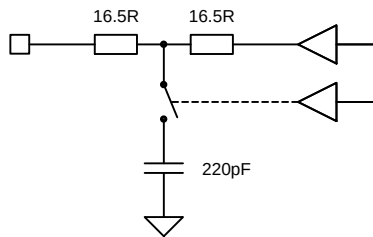


### TCK signal

The **TCK** output signal is similar to a standard output signal, but also has a switchable capacitor, forming a T-filter, which can reduce the **TCK** slew-rate.

Enabling this filter is not currently supported in Arm® Development Studio.

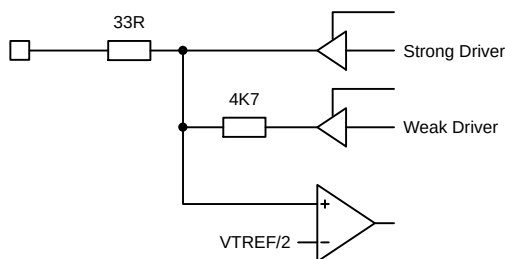
**Figure 2-13: TCK signal**



## Reset signals

The reset signals (**nSRST** and **nTRST**) are similar to the standard input/output signals. However, they have an extra LVCMOS driver, which is connected using a 4K7 resistor, that provides the weak pull-up and pull-down functionality.

**Figure 2-14: Reset signals**

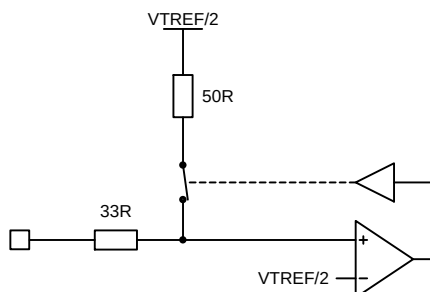


## Trace signals

The trace signals (**TRACEDATA[0-31]** and **TRACECLK**) are similar to standard inputs, but are also terminated to **VTREF/2** through 50Ω resistors. These resistors prevent signals from being reflected back to the target system, increasing signal integrity and the maximum data rate.

Disabling the input terminations is not currently supported in Arm Development Studio.

**Figure 2-15: Trace signals**



## VTREF signals

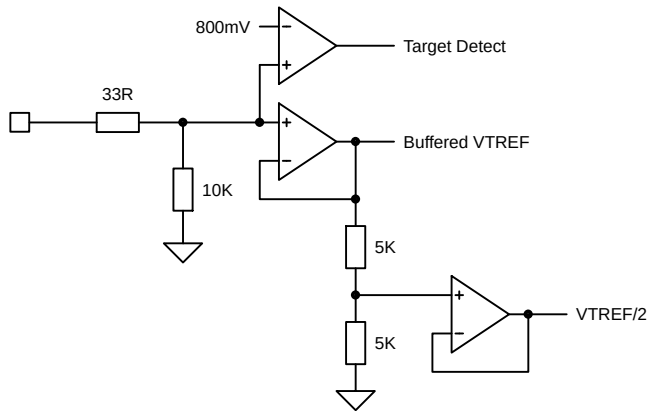
The **VTREF** signals (**VTREF**, **DEBUG\_VTREF** and **TRACE\_VTREF**) are buffered to provide:

- A VDD rail for the LVCMOS output buffers.

- The **VTREF/2** reference/termination rail.

For the debug unit to detect that a target is present, the **VTREF** signal must be higher than 800mV.

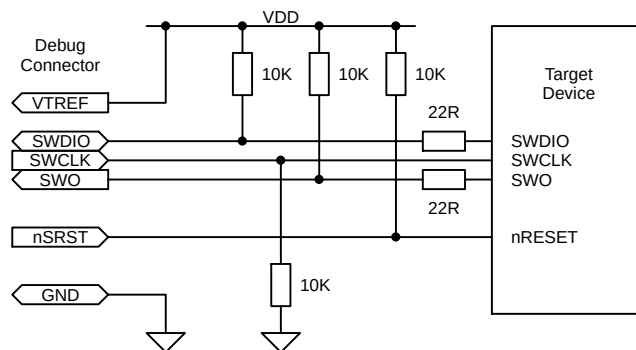
**Figure 2-16: VTREF signals**



## 2.9 Typical SWD circuit

A typical SWD circuit:

**Figure 2-17: Typical SWD circuit**

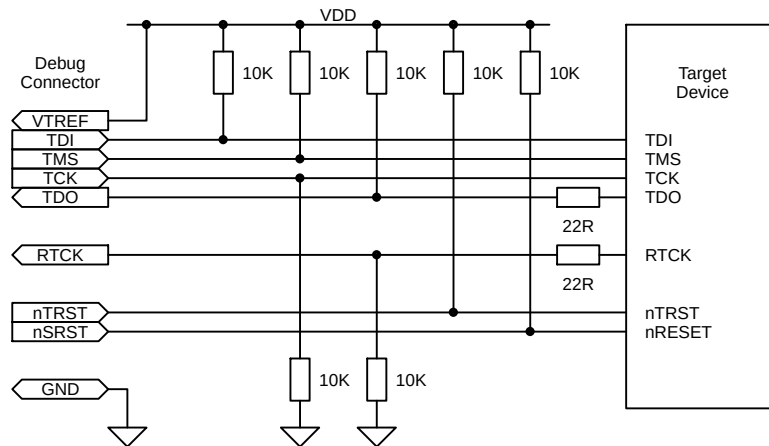


To improve signal integrity, it is good practice to provide an impedance-matching resistor on the **SWDIO** and **SWO** outputs of the processor. The value of these resistors, added to the impedance of the driver, must be approximately equal to 50Ω.

## 2.10 Typical JTAG circuit

A typical JTAG circuit:

**Figure 2-18: Typical JTAG circuit**



To improve signal integrity, it is good practice to provide an impedance matching resistor on the **TDO** and **RTCK** outputs of the processor. The value of these resistors, added to the impedance of the driver, must be approximately equal to 50Ω.

## 3. Target interface connectors

DSTREAM-ST has an Arm JTAG 20 connector, a CoreSight™ 20 connector, an auxiliary connector, and a user I/O connector.

The Parallel Trace probe provides an additional MIPI-60 connector.

To adapt debug connectors for other target connectors, you can use cables and adapter boards. Some of these cables and adapter boards are supplied with the debug unit. Others can be requested from Arm. For a list of provided adapters, see the [Arm DSTREAM-PT Getting Started Guide](#).

If your target system uses a connector which is not currently supported, and you are considering a volume order of Arm debug units, [contact Arm support](#) with your requirements. Arm might be able to supply a compatible adapter on a fast-turn, prototype basis.



All connector pinouts in this chapter are shown as they would appear on the target board.

### 3.1 Target connector selection guide

When choosing a debug or trace connector to design into a target board, there are many connector attributes to consider.

The connector attributes are:

**Table 3-1: Connector attributes**

Connector	Arm JTAG 20	CoreSight™ 10	CoreSight 20	TI JTAG 14	MICTOR 38	MIPI 34	MIPI 60
JTAG supported	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SWD supported	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SWO trace supported	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Parallel trace supported	No	No	Yes	No	Yes	Yes	Yes
Max parallel trace width <sup>1</sup>	N/A	N/A	4	N/A	16	4	32

<sup>1</sup> The trace width supported by the connector. DSTREAM-ST supports up to 4-bit wide parallel trace. DSTREAM-PT supports up to 32-bit wide parallel trace.



Connector	Arm JTAG 20	CoreSight™ 10	CoreSight 20	TI JTAG 14	MICTOR 38	MIPI 34	MIPI 60
Separate debug/trace voltage domains	No	No	No	No	Yes	Yes	Yes
Requires adapter	No	No	No	Yes	Yes	Yes	Yes
Cable signal integrity	Medium	Good	Good	Poor	Excellent	Good	Excellent
MIPI pinout compatible	No	Yes	Yes	No	No	Yes	Yes
Target connector cost	Low	Low	Low	Low	High	Low	Medium
Connector durability <sup>2</sup>	High	Low	Low	High	Medium	Low	Medium
Approximate footprint area (mm²) <sup>3</sup>	297	65	95	250	221	140	170
Through-hole/SMD	Either	Either	Either	Either	SMD <sup>4</sup>	Either	SMD
Ease of assembly (placement/soldering)	High	High	High	High	Low	High	Medium

## 3.2 Arm JTAG 20 connector

The Arm JTAG 20 connector is a 20-way 2.54mm pitch box header which supports JTAG debug, Serial Wire Debug, and SWO trace.



Caution

Using a non-shrouded header on the target board can lead to short-circuits or signal contention. To ensure the correct polarity and position, Arm recommends that you use a fully shrouded box header.

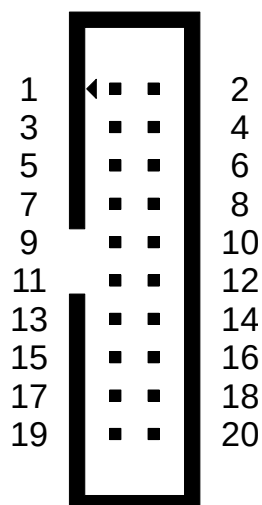
To use this connector with DSTREAM-ST, use the Arm JTAG 20 debug cable supplied in the box contents.

<sup>2</sup> Through-hole variants of connectors are more durable than SMD variants.

<sup>3</sup> Assumes an SMD part is being used. Through-hole parts use additional space on the opposite side of the board.

<sup>4</sup> The Mictor 38 connector is a hybrid part which has SMD signal pins and through-hole ground pins (which must be solder-pasted from the component side). Mictor 38 connector is not recommended for future designs.

**Figure 3-1: Arm JTAG 20 connector pinout**



**Arm JTAG 20 pinout table**

**Table 3-2: Arm JTAG 20 pinout table**

Pin	Signal name	Pin	Signal name
1	VTREF	2	NC
3	nTRST	4	GND
5	TDI	6	GND
7	TMS/SWDIO	8	GND
9	TCK/SWCLK	10	GND
11	RTCK	12	GND
13	TDO/SWO	14	GND
15	nSRST	16	GND
17	DBGRQ	18	GND
19	DBGACK	20	GND

## 3.3 CoreSight 10 connector

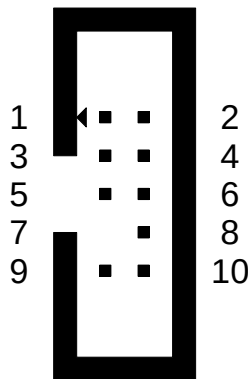
The CoreSight™ 10 connector is a 10-way 1.27mm pitch box header which supports JTAG debug, Serial Wire Debug, and SWO trace.



Caution

Using a non-shrouded header on the target board can lead to short-circuits or signal contention. To ensure the correct polarity and position, Arm recommends that you use a fully shrouded box header.

To use this connector with DSTREAM-ST, use the CoreSight 10/20 debug cable supplied in the box contents.

**Figure 3-2: CoreSight 10 connector pinout****CoreSight 10 pinout table****Table 3-3: Arm CoreSight 10 pinout table**

Pin	Signal name	Pin	Signal name
1	VTREF	2	TMS/SWDIO
3	GND	4	TCK/SWCLK
5	GND	6	TDO/SWO
7	Key (NC) <sup>5</sup>	8	TDI
9	GND	10	nSRST

## 3.4 CoreSight 20 connector

The CoreSight™ 20 connector is a 20-way 1.27mm pitch box header which supports JTAG debug, Serial Wire Debug, SWO trace, and up to 4-bit wide continuous-mode TPIU trace.

**Caution**

Using a non-shrouded header on the target board can lead to short-circuits or signal contention. To ensure the correct polarity and position, Arm recommends that you use a fully shrouded box header.

To use CoreSight 20 connector with the DSTREAM-PT system, use the CoreSight 20 debug cable supplied in the box contents.

You must configure the pinout mode of the CoreSight 20 connector before using it.

**Caution**

If you do not configure the CoreSight 20 connector correctly, your DSTREAM-PT system might not operate correctly. For example, if the pinout was configured for debug and trace, instead of debug only: the **nTRST** signal might terminate as if it

<sup>5</sup> Pin 7 must be removed for compatibility with DSTREAM-ST and MIPI specifications.

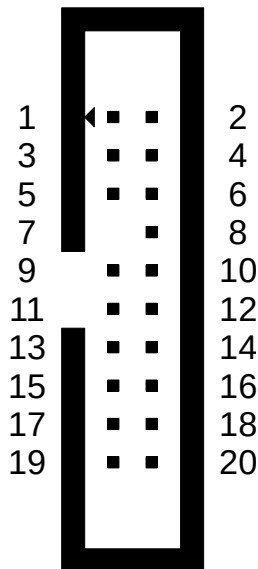
was a **TRACEDATA** signal, which causes the **nTRST** signal to be asserted, and might cause a TAP reset when the DSTREAM-PT system is connected.

To configure the pinout mode, use the [Platform Configuration Editor \(PCE\)](#) in Arm® Development Studio. In the PCE, select **Debug Adapter**, then select the **Probe Configuration** tab. In the configuration items table, set the `DSTREAMCS20` configuration item to either:

- 0: to use the connector in JTAG debug and trace mode.
- 1: to use the connector in JTAG debug only mode.

For more information, see [Configure your debug hardware unit for platform autodetection](#) in the Arm Development Studio User Guide.

**Figure 3-3: CoreSight 20 connector pinout**



### CoreSight 20 pinout tables

Pinout when `DSTREAMCS20` is set to 0:

**Table 3-4: Arm CoreSight 20 pinout table (`DSTREAMCS20=0`)**

Pin	Signal name	Pin	Signal name
1	VTREF	2	TMS/SWDIO
3	GND	4	TCK/SWCLK
5	GND	6	TDO/SWO
7	Key (NC) <sup>6</sup>	8	TDI
9	GND	10	nSRST

<sup>6</sup> Pin 7 must be removed for compatibility with DSTREAM-ST and MIPI specifications.

Pin	Signal name	Pin	Signal name
11	GND <sup>7</sup>	12	TRACECLK
13	GND <sup>7</sup>	14	TRACEDATA[0]
15	GND	16	TRACEDATA[1]
17	GND	18	TRACEDATA[2]
19	GND	20	TRACEDATA[3]

Pinout when DSTREAMCS20 is set to 1:

**Table 3-5: Arm CoreSight 20 pinout table (DSTREAMCS20=1)**

Pin	Signal name	Pin	Signal name
1	VTREF	2	TMS/SWDIO
3	GND	4	TCK/SWCLK
5	GND	6	TDO/SWO
7	Key (NC)	8	TDI
9	GND	10	nSRST
11	GND <sup>7</sup>	12	RTCK
13	GND <sup>7</sup>	14	SWO
15	GND	16	nTRST
17	GND	18	DBGREQ
19	GND	20	DBGACK

## 3.5 TI JTAG 14 connector

The Texas Instruments (TI) JTAG 14 connector is a 14-way 2.54mm pitch box header which supports JTAG debug, Serial Wire Debug, and SWO trace.

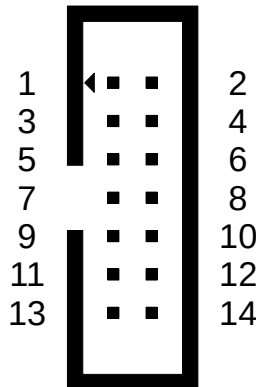


Caution

- Using a non-shrouded header on the target board can lead to short-circuits or signal contention. To ensure the correct polarity and position, Arm recommends that you use a fully shrouded box header.
- For the pin-out of the TI JTAG 14 connector, do not use 14-way IDC cables to connect the target board and debug unit. To avoid cross-talk issues between **nTRST**, **TMS**, and **TDI**, the DSTREAM-ST TI JTAG 14 adapter must connect directly to the target board.

To use this connector with DSTREAM-ST, the supplied TI JTAG 14 adapter must be used with the Arm JTAG 20 debug cable.

<sup>7</sup> Although pins 11 and 13 are typically grounded on the target board, the MIPI specification also allows them to carry power. If they are connected to a power rail (or rails) on the target board, these pins must also be AC-coupled to **GND**. To couple the pins to **GND**, use 100nF capacitors that are close to the connector.

**Figure 3-4: TI JTAG 14 connector pinout****TI JTAG 14 pinout table****Table 3-6: TI JTAG 14 pinout table**

Pin	Signal name	Pin	Signal name
1	TMS/SWDIO	2	nTRST
3	TDI	4	GND
5	VTREF	6	NC
7	TDO/SWO	8	GND
9	RTCK	10	GND
11	TCK/SWCLK	12	GND
13	DBGRRQ	14	DBGACK

## 3.6 Mictor 38 connector

The Mictor 38 connector is a 38-way 0.635mm pitch socket which supports JTAG debug, Serial Wire Debug, SWO trace, and up to 16-bit wide continuous-mode Trace Port Interface Unit (TPIU) trace.

**Caution**

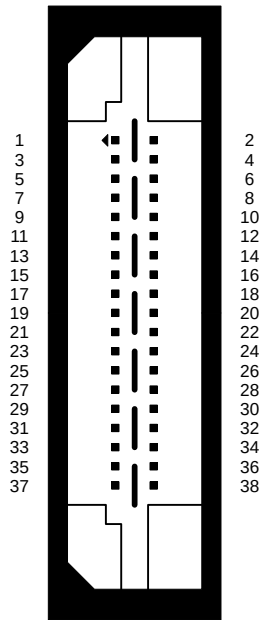
The center ground pins of the Mictor socket must be solder-pasted on the same side of the PCB as the connector. If you do not solder-paste on the same side of the PCB as the connector, it might cause mechanical or signal integrity issues.

Typically, the socket used is a 2-767004-2 from TE Connectivity.

To use this connector with DSTREAM-ST, the supplied 4-bit Mictor adapter must be used in conjunction with both the Arm JTAG 20 debug cable and the CoreSight™ 20 debug cable.

To use this connector with DSTREAM-PT, the supplied 16-bit Mictor adapter must be used in conjunction with the MIPI-60 cable.

**Figure 3-5: Mictor 38 connector pinout**



**Mictor 38 pinout table**

**Table 3-7: Mictor 38 pinout table**

Pin	Signal name	Pin	Signal name
1	NC	2	NC
3	NC	4	NC
5	GND	6	TRACECLK
7	DBGREQ	8	DBGACK
9	nSRST	10	EXTTRIG <sup>8</sup>
11	TDO	12	TRACE_VTREF <sup>9</sup>
13	RTCK	14	DEBUG_VTREF <sup>9</sup>
15	TCK	16	TRACEDATA[7]
17	TMS	18	TRACEDATA[6]
19	TDI	20	TRACEDATA[5]
21	nTRST	22	TRACEDATA[4]
23	TRACEDATA[15]	24	TRACEDATA[3]
25	TRACEDATA[14]	26	TRACEDATA[2]
27	TRACEDATA[13]	28	TRACEDATA[1]

<sup>8</sup> The **EXTTRIG** signal is deprecated and not supported by Arm Development Studio.

<sup>9</sup> Although the Arm CoreSight specification only supports a single **VTREF** (on pin 12), DSTREAM-ST can support separate debug and trace **VTREF**s. If only **TRACE\_VTREF** is powered, the DSTREAM-ST assumes that both debug and trace are to operate at that voltage.

Pin	Signal name	Pin	Signal name
29	TRACEDATA[12]	30	Logic 0 <sup>10</sup>
31	TRACEDATA[11]	32	Logic 0 <sup>10</sup>
33	TRACEDATA[10]	34	Logic 1 <sup>10</sup>
35	TRACEDATA[9]	36	TRACECTL <sup>11</sup>
37	TRACEDATA[8]	38	TRACEDATA[0]

## 3.7 Dual Mictor connectors

Two separate Mictor 38 connectors can be used to support JTAG debug, Serial Wire Debug, SWO trace, and up to 32-bit wide continuous-mode TPIU trace.



The center ground pins of the Mictor sockets must be solder-pasted on the same side of the PCB as the connector. If you do not solder-paste on the same side of the PCB as the connector, it might cause mechanical or signal integrity issues.

Typically, the sockets used are a 2-767004-2 from TE Connectivity.

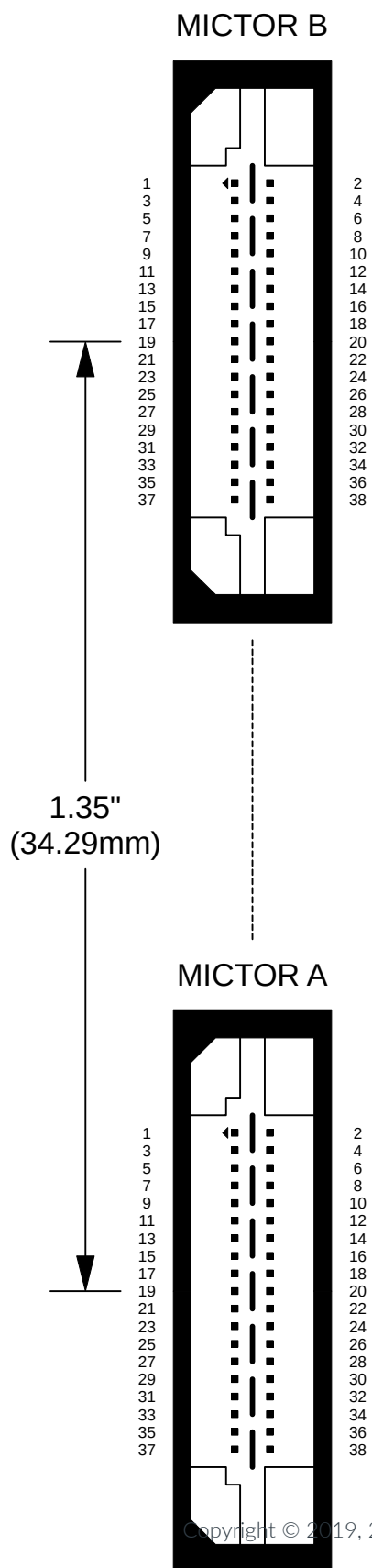
To use these connectors with DSTREAM-PT, the supplied 32-bit Mictor adapter must be used in conjunction with the MIPI-60 cable.

<sup>10</sup> **Logic 0** and **Logic 1** are not used by DSTREAM-ST. To maintain compatibility with other debug units, connect the signals to the appropriate power rails.

<sup>11</sup> The **TRACECTL** signal is not currently supported by DSTREAM-ST because only continuous-mode TPIU trace is supported.



**Figure 3-6: Dual Mictor connector pinout**



In this configuration, Mictor A is identical to the single [Mictor 38 connector](#).

Mictor B must be placed on a 34.29mm pitch above Mictor A and provides the extra signals necessary for 32-bit trace.

### Mictor B pinout table

**Table 3-8: Mictor B pinout table**

Pin	Signal name	Pin	Signal name
1	NC	2	NC
3	NC	4	NC
5	GND	6	NC
7	NC	8	NC
9	NC	10	NC
11	NC	12	TRACE_VTREF <sup>12</sup>
13	NC	14	NC
15	NC	16	TRACEDATA[23]
17	NC	18	TRACEDATA[22]
19	NC	20	TRACEDATA[21]
21	NC	22	TRACEDATA[20]
23	TRACEDATA[31]	24	TRACEDATA[19]
25	TRACEDATA[30]	26	TRACEDATA[18]
27	TRACEDATA[29]	28	TRACEDATA[17]
29	TRACEDATA[28]	30	Logic 0 <sup>12</sup>
31	TRACEDATA[27]	32	Logic 0 <sup>12</sup>
33	TRACEDATA[26]	34	Logic 1 <sup>12</sup>
35	TRACEDATA[25]	36	Logic 0 <sup>12</sup>
37	TRACEDATA[24]	38	TRACEDATA[16]

## 3.8 MIPI 34 connector

The MIPI 34 connector is a 34-way 1.27mm pitch box header which supports JTAG debug, Serial Wire Debug, SWO trace, and up to 4-bit wide continuous-mode TPIU trace.



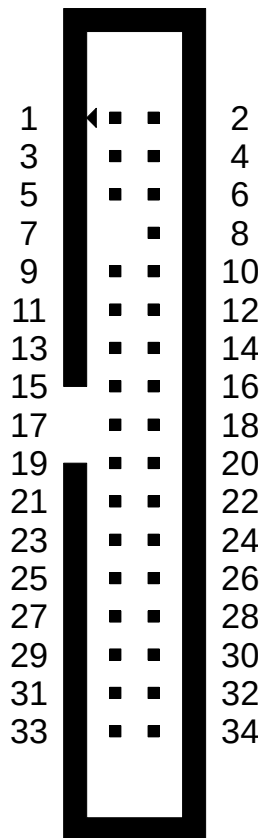
This connector is rarely used on target boards. The MIPI 34 adapter and debug cable is not supplied with the DSTREAM-PT unit, but are available on request.

<sup>12</sup> These signals are not used by Arm debug units. To maintain compatibility with other debug units, connect the signals to the appropriate power rails.



- Using a non-shrouded header on the target board can lead to short-circuits or signal contention. To ensure the correct polarity and position, Arm recommends that you use a fully shrouded box header.
- The MIPI 34 connector supports separate voltage domains for the debug and trace signals. You must supply the appropriate voltages to both of the **VTREF** pins.

**Figure 3-7: MIPI 34 connector pinout**



**MIPI 34 pinout table**

**Table 3-9: MIPI 34 pinout table**

Pin	Signal name	Pin	Signal name
1	DEBUG_VTREF	2	TMS/SWDIO
3	GND	4	TCK/SWCLK
5	GND	6	TDO/SWO
7	Key (NC) <sup>13</sup>	8	TDI
9	GND	10	nSRST

<sup>13</sup> Pin 7 must be removed for compatibility with DSTREAM-ST and MIPI specifications.

Pin	Signal name	Pin	Signal name
11	GND <sup>14</sup>	12	RTCK
13	GND <sup>14</sup>	14	TRST_PD <sup>15</sup>
15	GND	16	nTRST
17	GND	18	DBGRRQ
19	GND	20	DBGACK
21	GND	22	TRACECLK
23	GND	24	TRACEDATA[0]
25	GND	26	TRACEDATA[1]
27	GND	28	TRACEDATA[2]
29	GND	30	TRACEDATA[3]
31	GND	32	TRACEEXT <sup>16</sup>
33	GND	34	TRACE_VTREF

## 3.9 MIPI 60 connector

The MIPI 60 connector is a 60-way 0.5mm pitch socket which supports JTAG debug, Serial Wire Debug, SWO trace, and up to 32-bit wide continuous-mode TPIU trace.

Typically, the socket is a QSH-030-01-L-D-A from Samtec.



Note

- DSTREAM-PT supports capturing up to 32-bit wide trace (**TRACEDATA[0-31]**)
- To use this connector with DSTREAM-PT, use the MIPI debug cable supplied in the box contents.



Caution

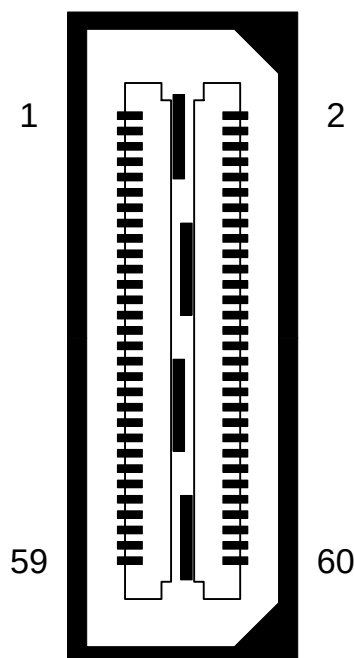
- The MIPI 60 connector supports separate voltage domains for the debug and trace signals. It is necessary to supply the appropriate voltages to both of the **VTREF** pins.
- Due to variations in naming conventions, you must take care when mapping the trace pins from the target device to the MIPI-60 connector. The table given here is correct for ETMv3 or later protocols.

<sup>14</sup> Although pins 11 and 13 are typically grounded on the target board, the MIPI specification also allows them to carry power. If they are connected to power rail (or rails) on the target board, these pins must also be AC coupled to **GND** using 100nF capacitors that are close to the connector.

<sup>15</sup> The **TRST\_PD** signal allows the target board to have a second TAP reset signal which is normally pulled-down. For more information, see the MIPI debug connector specification.

<sup>16</sup> The **TRACEEXT** signal is not supported by DSTREAM-ST.

**Figure 3-8: MIPI 60 connector pinout**



**MIPI 60 pinout table**

**Table 3-10: MIPI 60 pinout table**

Pin	Signal name	Pin	Signal name
1	DEBUG_VTREF	2	TMS/SWDIO
3	TCK	4	TDO
5	TDI	6	nSRST
7	RTCK	8	TRST_PD <sup>17</sup>
9	nTRST	10	DBGQRQ
11	DBGACK	12	TRACE_VTREF
13	TRACECLK	14	RESERVED <sup>18</sup>
15	GND	16	GND
17	TRACECTL <sup>19</sup>	18	TRACEDATA[19]
19	TRACEDATA[0]	20	TRACEDATA[20]
21	TRACEDATA[1]	22	TRACEDATA[21]
23	TRACEDATA[2]	24	TRACEDATA[22]
25	TRACEDATA[3]	26	TRACEDATA[23]
27	TRACEDATA[4]	28	TRACEDATA[24]
29	TRACEDATA[5]	30	TRACEDATA[25]

<sup>17</sup> The **TRST\_PD** signal allows the target board to have a second TAP reset signal which is normally pulled-down. For more information, see the MIPI debug connector specification.

<sup>18</sup> Pins marked as RESERVED might be internally connected in DSTREAM-ST, but are not currently supported.

<sup>19</sup> DSTREAM-ST ignores the TRACECTL pin since only continuous-mode (TPIU) trace is supported.

Pin	Signal name	Pin	Signal name
31	TRACEDATA[6]	32	TRACEDATA[26]
33	TRACEDATA[7]	34	TRACEDATA[27]
35	TRACEDATA[8]	36	TRACEDATA[28]
37	TRACEDATA[9]	38	TRACEDATA[29]
39	TRACEDATA[10]	40	TRACEDATA[30]
41	TRACEDATA[11]	42	TRACEDATA[31]
43	TRACEDATA[12]	44	RESERVED <sup>18</sup>
45	TRACEDATA[13]	46	RESERVED <sup>18</sup>
47	TRACEDATA[14]	48	RESERVED <sup>18</sup>
49	TRACEDATA[15]	50	RESERVED <sup>18</sup>
51	TRACEDATA[16]	52	RESERVED <sup>18</sup>
53	TRACEDATA[17]	54	RESERVED <sup>18</sup>
55	TRACEDATA[18]	56	RESERVED <sup>18</sup>
57	GND	58	GND
59	RESERVED <sup>18</sup>	60	RESERVED <sup>18</sup>

## 3.10 Auxiliary connector

The Auxiliary (AUX) connector on the front of DSTREAM-ST is used to support external probes for high-speed trace capture.



Note

- This connector is not intended for user I/O. Do not attempt to connect anything other than Arm® DSTREAM-ST compatible probes.
- This connector is not compatible with older RealView Trace (RVT) probes.

## 3.11 User I/O connector

To set up custom input or output signals to your target, use the user Input/Output (I/O) connector.



Caution

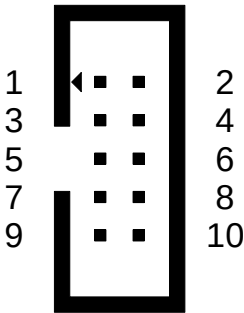
When connecting and disconnecting the user I/O port, Arm recommends that all equipment is powered-down.



- User outputs use the 3.3V LVCMOS standard and have a 100Ω series resistor for short-circuit protection.
- User inputs use the 3.3V LVCMOS standard and have a 10K series resistor and 100K pull-up resistor. You can safely drive the inputs up to a maximum of 5V.
- You can use the 3.3V power output to supply external circuitry up to a maximum current of 150mA. If an over-current condition occurs, this power output shuts down until the debug unit is reset.

The user I/O connector is a standard 10-way 2.54mm pitch box header on the rear of DSTREAM-ST.

**Figure 3-9: User I/O connector pinout**



**User I/O pinout table**

**Table 3-11: User I/O pinout table**

Pin	Signal name	Pin	Signal name
1	Output 1	2	Output 2
3	Output 3	4	Output 4
5	Output 5	6	Input 1
7	Output 6	8	Input 2
9	3.3V (output)	10	GND

## 4. Target board design

When you design a target board to connect to the DSTREAM-PT system, you must consider the rules that are described in this chapter.

### 4.1 Overview of high-speed design

When designing a target board that will be connected to a DSTREAM-PT system, it is important to use good digital design practice to achieve high *Signal Integrity* (SI).



A target system might work perfectly when it is connected to an older or slower debug unit, but it could fail to work with DSTREAM-ST because of the faster rise-times.

While many target boards already take SI into consideration for trace signals, it is also important to use the same design methodology for the debug signals. To achieve the high-data throughput that is required to debug modern target systems, DSTREAM-ST units are designed to drive their JTAG interfaces at up to 180MHz. To drive at this frequency, DSTREAM-ST units use fast output drivers with short rise-times.

There are many design rules that you can implement to ensure high SI in the debug and trace interfaces.

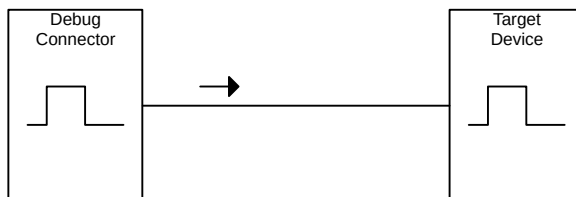


While these rules apply to all digital signals, Arm recommends giving special attention to the clock signals, **TCK**, **RTCK**, and **TRACECLK**.

#### Avoid stubs

Where possible, debug and trace signals should be point-to-point between the driver and receiver of the signal with no T-junctions or branches leading to other circuitry on the target board.

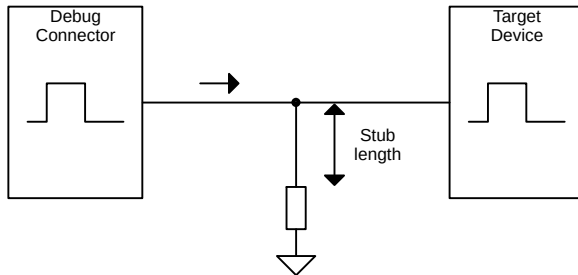
**Figure 4-1: Point-to-point signal**





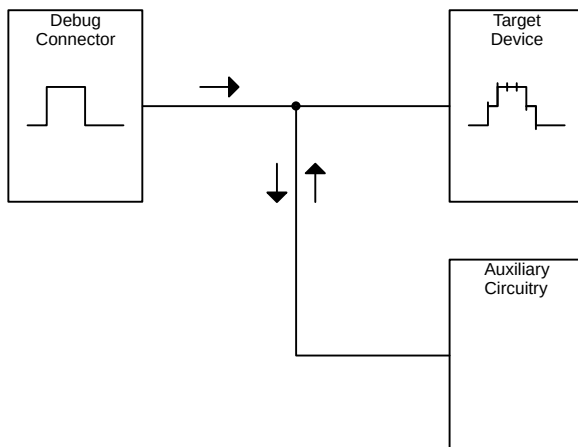
For debug signals, pull-up or pull-down resistors are often required. Pull-up or pull-down resistors might create a branch or stub in the signal path. It is important to keep the stub length in the signal path as short as possible.

**Figure 4-2: Stub length**



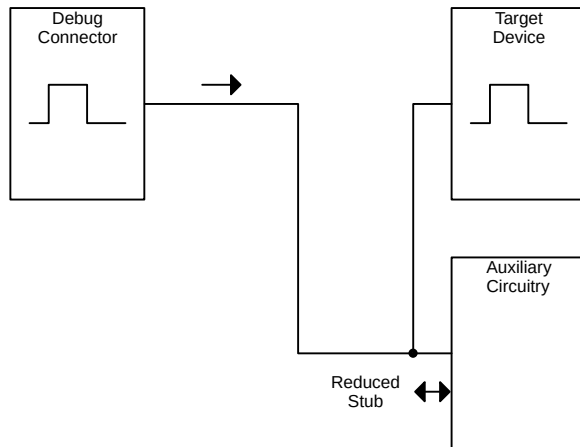
If a signal is routed with a long stub, the signal from the driver is split two ways when it reaches the T-junction. The signal that reaches the target device initially has a lower amplitude until the other half of the signal has reflected back from the end of the stub. The reflection has the effect of creating a stepped signal at the target device. A stepped signal at the target device can cause extra false signal edges to be received.

**Figure 4-3: Long stub causing false edges**



The simplest method to avoid a long stub causing false edges is to shorten the stub length by rerouting the signal. While rerouting the signal might add length to the signal route, the reduction in stub length is much more favorable.

**Figure 4-4: Improved route with shorter stub**



Alternative methods include:

- To prevent signal reflections affecting the incoming signal, use a buffer at T-junctions. This method is used to replicate clock signals.
- To route a signal without stubs, use an analog switch. This method is used when a device pin has multiple functions, for example, JTAG and general I/O.
- To deflect a larger portion of the signal away from the stub, use a resistor at the junction of the stub. This method is used when the stub leads to lower-bandwidth circuitry.

### Ensure the continuity of return signals

As a digital signal propagates along its route, an inverse signal travels through the adjacent plane because of the electric-field coupling between the signal and the plane. When the signal edges are short, the return signal usually follows the path of least inductance, rather than resistance. This means that the return signal flows through a path in the adjacent plane, that is as close as possible to the signal route. When the return path is interrupted, it causes distortion and some loss in the signal.

To minimize return path issues:

- Ensure that the return path that is adjacent to the signal is continuous with no slots or accidental voids that are caused by anti-pads.
- When routing a signal from one layer to another, link the planes close to the signal via using a return via. If the planes are at different voltages, use a low-value capacitor to AC-couple the return path.
- When routing signals to and from a cable connector, ensure that all of the return signals of the cable are used. Directly link the return signals or AC-couple them to ground, as necessary.

### Minimize crosstalk

Every signal route on a target board has some effect on nearby signal tracks because of the coupling of electric and magnetic fields between the tracks. The electric and magnetic field coupling causes small variations in the surrounding signals which, over long enough distances, can cause data corruption.

There are several ways to minimize electric and magnetic field coupling:

- Space the signal tracks further apart. Arm recommends to keep adjacent signals at least three times further apart than they are from the nearest plane (the 3W rule).
- Bring the plane closer to the signals. To reduce the 3W distance that is needed between adjacent signals, use thinner laminates between the signal and plane layers.
- Keep the signal tracks as short as possible. To cut down on routing while also reducing crosstalk, place a debug or trace connector closer to the target device.

### Use impedance matching

Every signal route on a target board has an effective impedance that is measured in Ohms. Effective impedance is the equivalent resistance to ground a signal experiences when it initially enters a signal route, before any reflection from the far end has occurred. It is important to note:

- If the different portions of a signal route have different impedances, it can cause reflections in the route. Reflections reduce the integrity of the signal.
- DSTREAM-ST is designed to work with target boards that use 50Ω signal tracks.

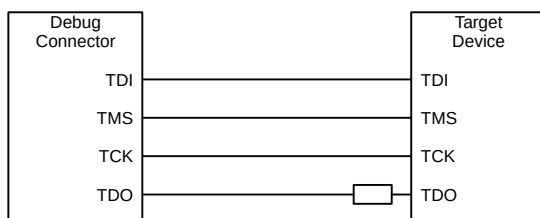
Most modern PCB design tools include functionality for calculating track impedance. There are also various free resources online for calculating the impedance of the various types of PCB track.

## 4.2 JTAG port buffering

JTAG buffering is sometimes required on the target board to improve signal integrity and increase the usable bandwidth of the interface. You can implement JTAG buffering using common off-the-shelf parts, at little cost.

Usually, the JTAG connector of a target system connects to a single device, for example:

**Figure 4-5: JTAG connection without buffers**

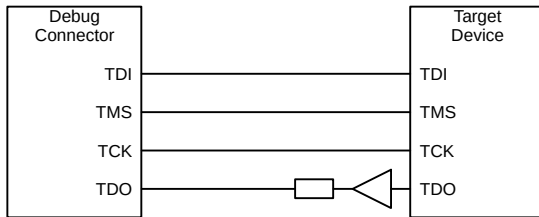


Pull-up and pull-down resistors are omitted for clarity.

To act as a series terminator, you must place a resistor close to the **TDO** pin of target device. Placing a resistor close to the **TDO** pin is the simplest option, and achieves good signal integrity because each signal is point-to-point.

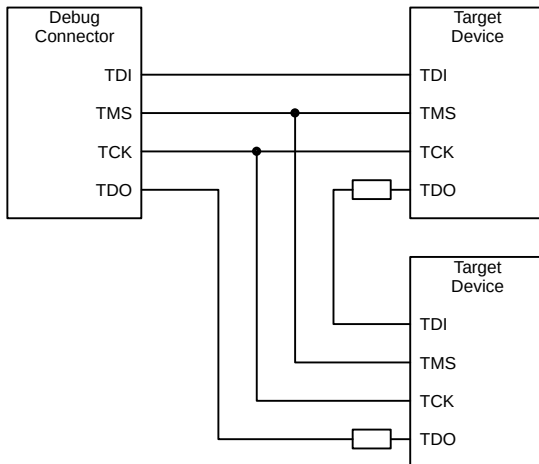
However, if the **TDO** output of the target device has a weak drive-strength (<4mA), the **TDO** output could significantly limit the maximum frequency of the JTAG interface. To resolve this, place a buffer close to the **TDO** pin of the target device with the appropriate series termination resistor:

**Figure 4-6: JTAG connection with TDO buffer**



Sometimes, two or more devices are chained together in the target system:

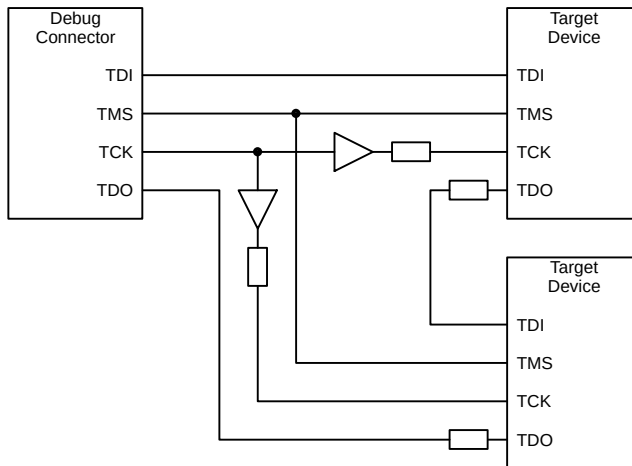
**Figure 4-7: Daisy-chained JTAG connection without buffers**



Achieving good signal integrity becomes difficult in this scenario because the **TMS** and **TCK** signals are branched at T-junctions. The signal integrity of the **TMS** signal is not important because until a rising edge of **TCK** signal is detected, it is ignored by the target device. The signal integrity of the **TCK** signal is important because any false edges cause the target device to sample **TDI** and **TMS** signals too many times. Sampling the **TDI** and **TMS** signals too many times corrupts the serial data stream that is seen by the target devices.

To avoid this issue, always use buffering where the **TCK** signal is split:

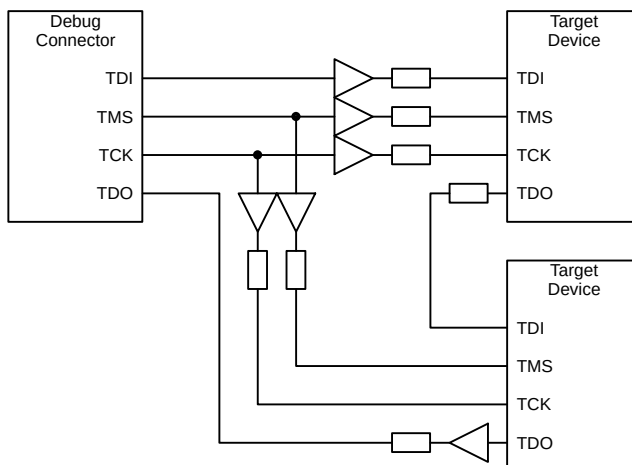
**Figure 4-8: Daisy-chained JTAG connection with TCK buffers**



The solution in the above figure prevents the two **TCK** branches from interacting and ensures good signal integrity with minimal overshoot. You must place buffers and series termination resistors as close as possible to the T-junction of the **TCK** signal.

This causes some skew between the **TDI**, **TMS**, and **TCK** signals. To correct this skew, use the same type of buffers on the **TDI**, **TMS**, and **TCK** signals. For example:

**Figure 4-9: Fully buffered JTAG connection**



This solution matches the skew between **TDI/TMS** and **TCK** signals to achieve high JTAG frequencies. Again, place the buffers and series termination resistors as close as possible to the T-junction of the **TMS** and **TCK** signals.



- For added noise rejection, Schmitt buffers can be used instead of standard buffers.
- Arm recommends you use buffers with a drive strength of 24mA or above.

- For any buffered signal, place the signal pull-up or pull-down resistor at the input-side of the buffer.

For guidance on selecting series termination resistors, see [Series termination](#).

## 4.3 Series termination

Series termination, or source termination, is a technique that is used in point-to-point signaling, to ensure that no excessive overshoot or ringing occurs.

To achieve series termination, use a series resistor to reduce the source voltage, by approximately 50%, as it is transmitted by the driver. When the signal reaches the end of the transmission line, the high impedance of the receiver causes a reflection that reverts the signal to its original amplitude. When the reflection returns to the series terminating resistor, the potential across the resistor drops to zero, preventing any further current from entering the transmission line. The receiver observes a perfect 100% logic transition, without any overshoot or ringing.

To ensure that a reliable signal is delivered to the DSTREAM-ST unit, Arm recommends that all outputs from the target system are simulated, and, if necessary, series terminated. Some overshoot or undershoot is acceptable, but Arm recommends ensuring this is kept less than ~0.5V. Above ~0.5V, the clamping diodes at the receivers start to cause high transient currents, which then cause increased crosstalk, radio emissions, and target power usage.

The target signal impedance for DSTREAM-ST is 50Ω.

When the outputs cannot be simulated, typical series terminating resistor values are:

**Table 4-1: Typical series terminating resistor values**

Driver strength	Typical series terminator	Notes
32mA	39Ω	Best signal integrity, highest speed
24mA	33Ω	-
16mA	27Ω	-
12mA	22Ω	-
8mA	15Ω	-
6mA	10Ω	Worst signal integrity, lowest speed

Some types of IC use *impedance matched* outputs to improve their signal integrity. Impedance matched outputs are commonly achieved by using weaker drive transistors to slow down the edge transitions. Using weaker drive transistors limits the data throughput of the driver.



Note

To achieve the highest data rates with the best signal integrity, Arm recommends using:

- A fast and strong driver.

- An appropriate series terminating resistor.

When series terminating multiple signals, it is common to use small quad resistor packages. Small quad resistor packages save board space, and reduce the parasitic effects without much risk of placement or tombstoning issues during production.



If you determine that series terminating resistors are not required, as a backup option, Arm strongly recommends that  $0\Omega$  links are placed close to the driver.

## 4.4 Parallel trace modeling

For trace bit rates of 0-600Mbps, basic signal integrity can be established using simplified modeling. Most of the transmission line model consists of the cable that is used to connect the DSTREAM-ST unit to the target.

- The 30cm CoreSight™ cable is made using 0.635mm pitch ribbon, and can be modeled as a  $66\Omega$  transmission line, with a 1.5ns propagation delay, and  $0.4\Omega$  DC resistance. The connectors at either end of the cable can be modeled as a 0.5pF capacitance to ground.
- The 15cm CoreSight cable is made using 0.635mm pitch ribbon, and can be modeled as a  $66\Omega$  transmission line, with a 0.75ns propagation delay, and  $0.2\Omega$  DC resistance. The connectors at either end of the cable can be modeled as a 0.5pF capacitance to ground.
- The JTAG 20 cable is made using 1.27mm pitch ribbon, and can be modeled as a  $100\Omega$  transmission line, with a 1.5ns propagation delay, and  $0.1\Omega$  DC resistance. The connectors at either end can be modeled as a 1.0pF capacitance to ground.
- The MIPI-60 cable is made using 0.5mm pitch micro-coaxial ribbon, and can be modeled as a  $50\Omega$  transmission line, with a 1.5ns propagation delay, and  $0.1\Omega$  DC resistance. The connectors at either end can be modeled as a 0.25pF capacitance to ground.

The circuit at the DSTREAM-ST end of the transmission line can be modeled using the following primitives:

- All resistors can be modeled as their ideal resistance values with minimum or zero parasitics.
- All capacitors can be modeled as their ideal capacitance values with minimum or zero parasitics.
- Input comparators can be modeled using the Spartan 3 SSTLx\_I model. The switching threshold can be assumed to be half of the VTREF voltage, as supplied by the target. The data is valid when it is 100mV above or below this threshold.
- Output drivers can be modeled using the Spartan 3 LVCMOS Fast 16mA model. You must choose the model voltage to match the target system voltage.

All other parasitics and traces within the DSTREAM-ST are negligible for most purposes.



To achieve good signal integrity, Arm recommends using series termination resistors on all target outputs.

## 4.5 Target design checklist

To ensure your target design is compatible with the DSTREAM-ST unit or DSTREAM-PT system, your answer to each applicable question in this checklist must be 'Yes'.



Not all questions are applicable to every target design.

**Table 4-2: Target design checklist**

Check item	Status
Are any <b>TDI</b> , <b>TMS</b> , <b>TDO</b> , or <b>SWDIO</b> signals pulled HIGH?	
Are any <b>TCK</b> , <b>RTCK</b> , or <b>SWCLK</b> signals pulled LOW?	
Are any <b>nTRST</b> or <b>nSRST</b> signals pulled to their inactive state (usually HIGH)?	
To pass data between the <b>TCK</b> domain and the internal clock domain, does the target device contain the necessary synchronization logic?	
If used, does <b>RTCK</b> have its own driver (separate from <b>TCK</b> )?	
If <b>TCK</b> is routed to multiple devices, have you used buffers to fan-out the signal (to prevent signal reflections)?	
Can the debug unit drive <b>nTRST</b> and <b>nSRST</b> separately?	
To allow debug from reset, can you reset the target device without initializing its debug logic?	
If using Serial wire Debug, is the <b>TMS/SWDIO</b> signal bidirectional (no uni-directional buffers)?	
To reduce the need to calibrate during setup, are any <b>TRACEDATA</b> and <b>TRACECLK</b> signals length-matched within a 10mm window?	
Where possible, have you eliminated stubs and other parasitic effects from debug and trace signals?	
To obtain 50Ω output impedance, have you routed any outputs from the target device through series termination resistors?	
Have the appropriate <b>VTREF</b> signal (or signals) been connected to the debug or trace connector (or connectors)?	
Either directly or through a resistor of 100Ω or less, are <b>VTREF</b> pins connected to the debug/trace logic rail (or rails)?	
Are the debug/trace logic rails in the range of 1.2V to 3.3V?	



Check item	Status
Are all GND pins of the debug/trace connector (or connectors) either directly connected, or AC-coupled, to GND, close to the connector?	
If using a Mictor socket, are the central GND pins solder-pasted on the same side of the board?	
If using dual Mictor sockets, are the connectors positioned with the correct spacing, orientation, and alignment?	
If using a standard 2.54mm or 1.27mm header, is the connector fully shrouded to avoid mis-connection (space permitting)?	
If using a CoreSight™ 10/20 or MIPI 34 connector, have you considered the removal of pin-7?	
To ensure the continuity of return paths, do any signal vias have return vias placed close to them?	
Have you checked the board layout to ensure that no signals cross slots or voids in the adjacent plane (or planes)?	
Where possible, has crosstalk between debug and trace signals been minimized?	
Are all debug and trace signals impedance-matched to 50Ω?	