



Help with debugging and tracing targets

Version 2.3

guide

Non-Confidential

Copyright © 2022–2023, 2025 Arm Limited (or its affiliates).
All rights reserved.

Issue 03

107551_2.3_03_en



Help with debugging and tracing targets guide

Copyright © 2022–2023, 2025 Arm Limited (or its affiliates). All rights reserved.

Release information

Document history

Issue	Date	Confidentiality	Change
0200-03	23 April 2025	Non-Confidential	Added more resources
0200-02	1 March 2023	Non-Confidential	Added more resources
0200-01	16 September 2022	Non-Confidential	Removes references to DS-5 Development Studio
0100-01	22 August 2022	Non-Confidential	Initial release

Proprietary Notice

This document is protected by copyright and other related rights and the use 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 Limited ("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 the subject matter of this document infringes any third party patents.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm's view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. 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, any patents, copyrights, trade secrets, trademarks, or other rights.

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.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted 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 this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.

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. Please follow Arm's trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

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.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Contents

1. Overview.....	7
2. I have a platform configuration but I cannot connect to my target with Arm DS. Why?.....	8
3. Processors are in a powered down state.....	10
4. Processors are held in reset.....	11
5. OS is interfering with the debugger.....	12
6. Target is not set up for debug.....	13
7. Reset signal strength used by the debug probe is not correct for the target.....	14
8. System is unstable.....	15
9. Debug connector or signal issues.....	16
10. Debug probe or debug cable issues.....	17
11. I can connect to my target, but I am not getting trace data or I am seeing corrupted trace data. Why?.....	18
12. Initial trace capture did not capture any or enough data.....	20
13. Trace devices are not listed in the platform configuration SDF file.....	21
14. Component connection was not visible through PCE testing.....	22
15. Incorrect trace infrastructure.....	23
16. Target is not fully set up for trace.....	24
17. Discrepancy between possible and working trace port width.....	25
18. No or unstable trace clock.....	26
19. Trace clock signal is too fast.....	27

20. Working with DSTREAM-ST.....	28
21. Working with DSTREAM-PT.....	29
22. Working with DSTREAM-HT.....	30
23. Working with DSTREAM-XT.....	31
24. Trace signal or connector issues.....	32
25. Debug probe or debug or trace cable issues.....	33
26. I need help debugging my target. Where do I go?.....	34
27. I am working with a certain target. Are there existing resources to help me get started?.....	35

1. Overview

Learn about common questions and answers when debugging or tracing a target with Arm Development Studio (Arm DS). This page focuses on connecting to, debugging, and tracing emulation, FPGA, and silicon targets.

To debug a target with Arm DS, you must have the following:

- A debug probe, like DSTREAM, [DSTREAM-ST](#), [DSTREAM-PT](#), [DSTREAM-HT](#), [DSTREAM-XT](#), and [ULINKpro family](#) debug probes, or a Functional I/O connection.
 - For more information, see the [What debug probes or connection types can I use to autodetect and connect to my target?](#) section of [Help with connecting to new targets guide](#).
- An Arm DS installation that supports the processor you want to debug.
 - Read [Supported Devices](#) for information on the processor cores Arm DS supports.
- An Arm DS license that enables your Arm DS edition.

If you cannot find solutions to your questions here:

- Read [I need help debugging my target. Where do I go?](#)
- Visit the [Arm Community](#)

Read the following sections for help debugging and tracing with Arm Development Studio (Arm DS):

- [I have a platform configuration but I cannot connect to my target with Arm DS. Why?](#)
- [I can connect to my target, but I am not getting trace data or I am seeing corrupted trace data. Why?](#)

Read the following sections for help working with a target:

- [I need help debugging my target. Where do I go?](#)
- [I am working with a certain target. Are there existing resources to help me get started?](#)

2. I have a platform configuration but I cannot connect to my target with Arm DS. Why?

Unfortunately, you can have an Arm Development Studio [platform configuration](#), but still not successfully connect to your target. Many factors can prevent a successful debug connection.

The following is a list of some common reasons why a debug connection is unsuccessful:

- [Processors are in a powered down state](#)
- [Processors are held in reset](#)
- [OS is interfering with the debugger](#)
- [Target is not set up for debug](#)
- [Reset signal strength used by the debug probe is not correct for the target](#)
- [System is unstable](#)
- [Debug connector or signal issues](#)
- [Debug probe or debug cable issues](#)

The following is a list of resources to help diagnose debugger connection issues:

- [Debug over power-down](#) blog
 - General information about processor power down and how it can affect a debugger.
- [Initializing a target section of the Learn the Architecture - Before debugging on Armv8-A Developer guide](#)
 - Guide section that covers extra steps to get a debugger connection
- [Target state section of the Learn the Architecture - Before debugging on Armv8-A Developer guide](#)
 - Guide section that covers possible target states on debugger connection.
- [Debugging over powerdown section of the Learn the Architecture - Debugger usage on Armv8-A Developer Guide](#)
 - Guide section that covers how powerdown effects debug capabilities in Armv8-A systems.
- If you are using Arm Development Studio 2019.0 or later, [Coresight Access Tool for SoC600](#).
 - CSAT600 is a low-level access tool used with [ARM Debug Interface Architecture Specification ADIV6.0](#) compliant boards. CSAT600 tests particular aspects of a [CoreSight SoC-600](#) debug infrastructure design.
- [CoreSight Access Tool \(CSAT\)](#) material
 - CSAT is a low-level access tool used with [ARM Debug Interface Architecture Specification ADIV5.0 to ADIV5.2](#) or earlier compliant boards. CSAT tests particular aspects of a [CoreSight](#) debug infrastructure design.

- [Debugging Armv8 platforms with CSAT tutorial](#)
- [Low Level Debug using CSAT on Armv7-based Platforms tutorial](#)
- [How to use the DAP logger tool KBA](#)
 - Arm Development Studio ships with a logging tool that captures detailed low-level logs of CoreSight systems. This logging tool works with DSTREAM, [DSTREAM-ST](#), [DSTREAM-PT](#), [DSTREAM-HT](#), and [DSTREAM-XT](#). The preceding KBA gives instructions on how to enable the DAP logger.
- [How do I get a DAP log when using ULINK family, third-party, or low cost debug probes? KBA](#)
 - Arm Development Studio ships with the ability to capture detailed low-level logs of CoreSight systems. This debug log works with [ULINKpro family](#) units. The preceding KBA gives instructions on how to capture a debug log.

3. Processors are in a powered down state

On some targets, processors or cores are powered down. If a core is powered down on connection, Arm Development Studio displays a `< core_number > powered down` message in the Debug Control view. If a processor or core is powered down, connecting to the processor or core or performing debug operations, like run, stop, and step, might fail.

Steps:

- Read [What does “Processor powered down” mean?](#)

4. Processors are held in reset

On some targets, processors or cores are held in reset. If a processor or core is held in reset, performing debug operations, like run, stop, and step, is not possible. If a core is in reset on connection, Arm Development Studio displays `< core_number > held in reset` in the Debug Control view.

Steps:

- Read [JTAG Reset Strenght and Timing](#).

5. OS is interfering with the debugger

Depending on the target, an OS might be running on the target when a debug connection is attempted. Some OSes prevent debug connections or cause unexpected debugger behavior.

The following are common items to check for when debugging a target with an OS running:

- The OS has not powered down devices necessary for debug.
- The OS has enabled the device clock source.
- The OS has configured the device reset correctly.
- The platform management controllers allow debug access.
- If extra drivers are necessary to manage or register target resources.
- The OS was built with debug support.

Steps:

- Consult the OS and target documentation if any of the previous items are interfering with debugging your target.
 - If debugging a Linux kernel, read [Why are Arm Development Studio or DS-5 unable to control my target when debugging the Linux kernel via JTAG?](#)
 - If debugging an Android native library, read [Arm DS/DS-5 is showing gdbserver errors when I try to debug my Android native library.](#)
- If you are encountering issues using software breakpoints when debugging Linux, read [Why does Arm DS or DS-5 software breakpoint not work for Linux kernel debug?](#)
- If you are encountering issues when booting Linux on a NXP i.MX6 or NXP i.MX7 target, read [Why does Linux fail to boot on NXP i.MX6 or i.MX7 when Arm DS is connected?](#)

6. Target is not set up for debug

Some targets might not support debug connections out-of-the-box and might require extra steps to make a debug connection possible.

The following are common extra steps to check for:

- If you are working with an FPGA target, a debug-enabled image is loaded. Also, the correct hardware components, like switches and cards, are used.
- Debug connectors are enabled.
- If debug and trace signals are shared with other devices, like General Purpose Input/Output (GPIO).

Steps:

- Consult the target designer, manufacturer, or documentation to determine whether the target is set up for debugging.
 - If you have a target example where debug is enabled, try connecting to the example first. A successful debug connection to the target example proves that the target can be configured for debug.

7. Reset signal strength used by the debug probe is not correct for the target

If autodetection does not complete successfully, the reset signal strength might be incorrect for the target. Try adjusting the debug probe reset signal strength.



Note

Some targets do not use the nSRST and nTRST reset signals. If the nSRST and nTRST signals are not used, check that these signals are tied off according to your debug probe documentation. For more information about the nSRST and nTRST signals, read the following:

- [Do I need to connect nSRST to the JTAG connector?](#)
- [Do I need to connect nTRST to the JTAG connector?](#)

Steps:

- Read [JTAG Reset Strength and Timing](#).

8. System is unstable

If the system you have connected to is in an unstable state, the debugger connection might fail. Some examples of an unstable system are the core is in a fault or abort start or PCE autodetection fails for core-specific reasons. If the debug connection fails because of an unstable system, after connecting, a `Failed to obtain target state` message might appear in the Commands view.

Steps:

- Read [Effects of an Unstable Debugger Connection](#).
- Read [How do I use DS-5 or Arm DS to debug issues when a CPU hangs?](#)

9. Debug connector or signal issues

Debug connectors or the debug signals might not work as expected. Problems with debug connectors or signals can prevent Arm Development Studio from connecting to a target or cause unstable debug connections.

Steps:

- Read [Debug Connector Issues](#).

10. Debug probe or debug cable issues

Target autodetection is reliant on having a working debug probe and debug cables. Also, the debugger must be able to connect to and communicate with the debug probe. If the debug probe or cable setup is not working with other targets, you might have a broken debug setup.

Steps:

- If you cannot browse for your DSTREAM debug probe, read the following resources:
 - [What should I do if my DSTREAM family debug probe is not selectable in Arm Development Studio?](#)
 - [TCP and UDP usage for DSTREAM family units](#)
 - [Obtaining IP address of DSTREAM connected via USB](#)
- Read [I have an Arm DS or DS-5 platform configuration. Why can I not connect to my board?](#)
- If you are using a DSTREAM-ST debug probe, read the [Troubleshooting section of the DSTREAM-ST Getting Started Guide](#).
- To diagnose issues using the LEDs on the top of the DSTREAM-ST debug probe, read the [Indicator LEDs on the top section of the DSTREAM-ST Getting Started Guide](#).
- If you are using a DSTREAM-XT debug probe, read [Troubleshooting section of the Arm DSTREAM-XT Getting Started Guide](#).
- If you are using an ST-LINK debug probe from STMicroelectronics, read [Problems connecting to target hardware using Arm Development Studio and the ST-LINK debug probe](#).

11. I can connect to my target, but I am not getting trace data or I am seeing corrupted trace data. Why?

Unfortunately, a successful debug connection does not mean trace data appears automatically or entirely valid trace data is captured. If the trace data is corrupted, a `corrupted` message appears in the Arm Development Studio Trace view.

The following are common reasons why trace data might not appear or might be corrupted:

- [Initial trace capture did not capture any or enough data](#)
- [Trace devices are not listed in the platform configuration SDF file](#)
- [Component connection was not visible through PCE testing](#)
- [Incorrect trace infrastructure](#)
- [Target is not fully set up for trace](#)
- [Discrepancy between possible and working trace port width](#)
- [No or unstable trace clock](#)
- [Trace clock signal is too fast](#)
- [Working with DSTREAM-ST](#)
- [Working with DSTREAM-PT](#)
- [Working with DSTREAM-HT](#)
- [Working with DSTREAM-XT](#)
- [Trace signal or connector issues](#)
- [Debug probe or debug or trace cable issues](#)

The following are resources to help you with various trace situations:

- To test if your trace configure is capable of generating any valid trace data, try the following method:
 - There are two ways to capture trace: off-chip or on-chip.
 - With off-chip trace capture, the trace data is output from the target to a debug probe or directly to the external debugger.
 - With on-chip trace capture, the trace data is on-chip and is exported to the external debugger.
 - If off-chip trace capture, like tracing with a TPIU, is not working as expected, switch to using on-chip trace capture. Using on-chip trace capture, like Embedded Trace Buffer (ETB), Embedded Trace Funnel (ETF), Embedded Trace Router (ETR), or Trace Buffer Extension (TRBE), might give you a better trace result.

- If you are using off-chip trace capture, to check if the trace signals from the target are correct and valid, read [How to test TPIU trace signals](#).
- When using a target with an [Embedded Trace Macrocell v4 \(ETMv4\)](#) implementation, if you are seeing trace data but the cycle count values are unexpected or difficult to understand, read [Understanding generated ETMv4 cycle count values](#).
- If your target has an Embedded Trace Extension (ETE) and a TRBE and you want help decoding the generated trace data, read [Scripts to decode ETE trace stored to a TRBE](#).

12. Initial trace capture did not capture any or enough data

When tracing, no relevant trace data or not enough trace data is captured for any data to be displayed in the Trace view. Try running the target again for a longer period of time and see if any trace data appears.

13. Trace devices are not listed in the platform configuration SDF file

The [Platform Configuration Editor \(PCE\)](#) collects information about the trace devices behind the DAP to determine which trace devices are present. The information is collected by reading specific device Debug registers. If the device Debug registers are not accessible, the trace device is not added to the platform configuration. If the Debug registers for a trace device are not accessible, in the PCE Console view, a `Failed to read x bytes from AP < AP number > @ < component debug register base address >` message appears.

The following lists common reasons why the Debug registers are not accessible:

- The device is powered down.
- The device cannot enter Debug state. The inability to enter Debug state might occur because of a stall or lock up on the memory bus.
- Debug capability is disabled. Software running on the target, like an OS, can disable debug.

Steps:

- Read [How PCE identifies the CoreSight components on the target board](#).
- Read [Common reasons why components and component connections do not appear](#).
- Read [Can I visualize a SoC debug, trace, and CTI topology?](#)

14. Component connection was not visible through PCE testing

The Platform Configuration Editor (PCE) interrogates the Debug registers of a component to determine what is connected to the component.

The following are common reasons why the component connections are not found:

- The tested component is powered down.
- The connected components are powered down.
- Debug capability is disabled. Software running on the target, like an OS, can disable debug.

Steps:

1. Determine why the PCE component connection was not found and fix the issue. To help determine why the component connection is not found, read [Can I visualize a SoC debug, trace, and CTI topology?](#), [Common reasons why components and component connections do not appear](#), and [How PCE identifies the CoreSight components on the target board](#).
2. Try the autodetection process again.
3. If the reason for the missing component connection is not determinable, manually add a component connection to the platform configuration .sdf file. For instructions on how to add a component connection, read [Manually configuring a platform configuration for trace section of the Arm Debugger Manual Configuration Tutorial](#).

15. Incorrect trace infrastructure

In a platform configuration, some .sdf files might have incorrect trace component connections because of hardware issues or incorrectly connected components. For example, a trace source is connected to the wrong trace funnel port in the .sdf file. Another example, a ETMv4 trace source with both instruction and data trace only has one trace funnel port assigned to it instead of two.

Steps:

- Read [Possible effects of having an incorrect trace infrastructure](#).
- Read [Can I visualize a SoC debug, trace, and CTI topology?](#)

16. Target is not fully set up for trace

Some boards might not support trace connections out-of-the-box and require extra steps to make a trace connection possible.

Common extra steps to check for are:

- If you are working with an FPGA target, the debug and trace-enabled image is loaded. Also, the correct hardware components, like switches and cards, are used.
- Debug and trace connectors are enabled.
- If the target has a separate trace connector, the trace cable is plugged in.
- If there are multiple debug or trace connectors on the target, the correct trace connector is used.
- Trace signals are not output to a physical connector.
- Trace signals are shared with other devices, like General Purpose Input/Output (GPIO).
- The necessary trace devices are not powered up.

Steps:

- Consult the target designer, manufacturer, or documentation to determine whether the target is set up for tracing.
 - If you have a target example where trace is enabled, try connecting to the example first. A successful trace connection to the target example proves that the target can be configured for trace.
- If you are not seeing timestamps in the decoded trace data, read [How do I enable timestamps in my trace?](#)

17. Discrepancy between possible and working trace port width

Target documentation states the off-chip trace capture port width of the target. The off-chip trace capture port width is often referred to as the TPIU port width. When executing real-world code, you might find:

- The effective trace port width is less than what is stated in the target documentation. For example, if the trace port width is too large, you might see no or corrupted trace in the Trace view.
- The trace port width does not support the amount of trace data being generated. For example, if the trace port width is too small, you might see `Buffer overflow` messages in the Trace view.

Steps:

- Read [External Trace Port Width and Bandwidth](#).
- If trace is coming out of the Trace Port Interface Unit (TPIU) of your target, read the following:
 - [How to use the 'autocalibrate' tool to enable accurate off-chip trace capture](#)
 - [How to test TPIU trace signals](#)
 - [How can I add trace clock or signal delays when tracing to TPIU on Arm DS?](#)

18. No or unstable trace clock

You must have a valid, stable trace clock signal to capture off-chip trace data. Off-chip trace is where trace data generated on an SoC is passed to a debug probe and then to an external debugger. Off-chip trace is often referred to as Trace Port Interface Unit (TPIU) trace.

For a DSTREAM, [DSTREAM-ST](#), or [DSTREAM-PT](#), the TRC CLK LED indicates the state of the trace clock signal. If the TRC CLK LED is solid green, the DSTREAM family debug probe has registered a valid, stable trace clock signal. If the TRC CLK signal is not on, flashing red, or solid red, there is not a valid, stable trace clock signal.

For [DSTREAM-HT](#), the TRC CLK LED only applies when you are using parallel trace. The status of the TRC CLK LED does not apply when performing HSSTP trace.

With [DSTREAM-XT](#), trace data is acquired through the PCIe interface of the target. For help configuring your DSTREAM-XT for trace, read [Configuring your DSTREAM-XT for trace over PCIe](#). To learn the meanings of the trace LEDs on the DSTREAM-XT probe, read [The DSTREAM-XT probe section of the DSTREAM-XT Getting Started Guide](#).

Steps:

- If using DSTREAM, DSTREAM-ST, or DSTREAM-PT, read the following:
 - [TRC CLK LED flashing red, or off when tracing to DSTREAM unit](#)
 - [How to use the 'autocalibrate' tool to enable accurate off-chip trace capture](#)
 - [How can I add trace clock or signal delays when tracing to TPIU on Arm DS?](#)
- If using DSTREAM-HT, read [Working with DSTREAM-HT](#).
- If using DSTREAM-XT, read [Working with DSTREAM-XT](#).

19. Trace clock signal is too fast

On some targets, the trace clock signal might be running at too high a frequency for the rate the trace data is appearing. A high trace clock frequency can cause trace buffer overflows that lead to corrupted trace data. If the trace clock frequency is too high, you might see `Buffer overflow` or `corrupted` messages in the Trace view.

Steps:

- Read [What is the maximum frequency of debug and trace clocks in a CoreSight design?](#)
- If using DSTREAM, DSTREAM-ST, or DSTREAM-PT, read the following:
 - [TRC CLK LED flashing red, or off when tracing to DSTREAM unit.](#)
 - [How to use the 'autocalibrate' tool to enable accurate off-chip trace capture](#)
 - [How can I add trace clock or signal delays when tracing to TPIU on Arm DS?](#)

20. Working with DSTREAM-ST

The [DSTREAM-ST](#) supports capturing narrow-port (4-pin) parallel streaming trace from a target. If the DSTREAM-ST or target are not configured correctly for trace, no trace data or corrupted trace data might occur.

Steps:

- Read the [Troubleshooting section of the DSTREAM-ST Getting Started Guide](#).

21. Working with DSTREAM-PT

The [DSTREAM-PT](#) has a Real-Time Monitor (RTM) that calibrates the trace signals during trace capture. The RTM allows trace data to be captured more reliably. The RTM might fail to get a lock on the trace signals for signal calibration. If the RTM cannot get a signal lock, you might not see trace data in the Trace view.

Steps:

- Read [Troubleshooting DSTREAM-PT Trace](#).

22. Working with DSTREAM-HT

Capturing HSSTP trace requires many steps to set up the target. It is easy to miss required steps when setting up or working with the [DSTREAM-HT](#).

Steps:

- Read [Troubleshooting section of the Arm DSTREAM-HT Getting Started Guide](#).

23. Working with DSTREAM-XT

Capturing trace using the PCIe interface on a target requires knowledge of the PCIe interface implementation. You might encounter problems getting a working trace connection between the [DSTREAM-XT](#) and the target.

Steps:

- Read [Configuring your DSTREAM-XT for trace over PCIe](#).
- Read [Troubleshooting section of the Arm DSTREAM-XT Getting Started Guide](#).

24. Trace signal or connector issues

Sometimes the trace connector or the trace signals are not working as expected. Problems with trace connectors or signals can cause no trace or invalid trace to be shown in the Trace view.

Steps:

- Read [Trace Signal or Trace Connector Issues](#).
- If trace is coming out of the Trace Port Interface Unit (TPIU) of your target, read the following:
 - [How to use the 'autocalibrate' tool to enable accurate off-chip trace capture](#)
 - [How to test TPIU trace signals](#)
 - [How can I add trace clock or signal delays when tracing to TPIU on Arm DS?](#)

25. Debug probe or debug or trace cable issues

Target autodetection is reliant on having a working debug probe and debug cables. Also, the debugger must be able to connect to and communicate with the debug probe. If the debug probe or cable setup is not working with other targets, you might have a broken debug setup or trace setup.

Steps:

- If you cannot browse for your DSTREAM debug probe, read the following resources:
 - [What should I do if my DSTREAM family debug probe is not selectable in Arm Development Studio?](#)
 - [TCP and UDP usage for DSTREAM family units](#)
 - [Obtaining IP address of DSTREAM connected via USB](#)
- Read [I have an Arm DS or DS-5 platform configuration. Why can I not connect to my board?](#)
- If you are using a DSTREAM-ST debug probe, read the [Troubleshooting section of the DSTREAM-ST Getting Started Guide](#).
- To diagnose issues using the LEDs on the top of the DSTREAM-ST debug probe, read the [Indicator LEDs on the top section of the DSTREAM-ST Getting Started Guide](#).
- If you are using a DSTREAM-XT debug probe, read [Troubleshooting section of the Arm DSTREAM-XT Getting Started Guide](#).
- If you are using an ST-LINK debug probe from STMicroelectronics, read [Problems connecting to target hardware using Arm Development Studio and the ST-LINK debug probe](#).
- If you are using a Mictor 38 connector with DSTREAM-ST, read [Why do I need to connect both cables to the DSTREAM-ST Mictor adapter?](#)

26. I need help debugging my target. Where do I go?

If you need help with debugging or tracing your target with Arm Development Studio, read the following:

[Requesting help with board bring-up](#)

27. I am working with a certain target. Are there existing resources to help me get started?

The following links provide information about bringing up certain targets with Arm Development Studio:

- [Arm Development Studio Heterogeneous system debug with Arm DS](#)
 - An Arm DS document on how to set up an NXP i.MX7 SABRE development board for Cortex-A Linux and Cortex-M bare-metal application debug.
- [How to enable and use CMSIS-DAP debug on the NXP Semiconductors Vybrid Controller Tower System module](#)
 - A tutorial on how to update the OpenSDA firmware and debug an NXP Semiconductors Vybrid Controller Tower System module.
- [DS-5 support with the HiKey 960 board](#)
 - A blog on connecting to a Hikey 960 board.
- [Getting started with DS-5 and i.MX7](#)
 - A blog on connecting to an i.MX7 board.
- [Heterogeneous development made easy: STM32MP1 device support within Arm development tools](#)
 - A blog on working with STM32MP1 series devices with Arm Development Studio and Keil MDK.
- [Getting started with Cortex-M using Arm DS and CMSIS](#)
 - A KBA describing how to connect to a Cortex-M target with CMSIS in Arm Development Studio.
- [RD-N2 and RD-V2 Debug Tools Reference](#)
 - A KBA of a collection of references for debug tools and techniques that are useful for debugging and bringing up an RD-V2 or RD-N2 based system.
- [Can't connect Arm DS to NXP i.MX8MPlus board](#)
 - A KBA to help connect to the NXP i.MX8MPlus board with Arm Development Studio.
- [Why does Linux fail to boot on NXP i.MX6 or i.MX7 when Arm DS is connected?](#)
 - A KBA that provides help if your development board boots Linux normally when the debugger is not connected, but fails to boot when the debugger is connected.
- [Getting Started with the Ultrascale+ and Arm DS guide](#)
 - A guide on how to connect to and get code running on the Xilinx UltraScale+ and 96 Boards Ultra96 targets using Arm DS.