



# Help with connecting to new targets

Version 1.0

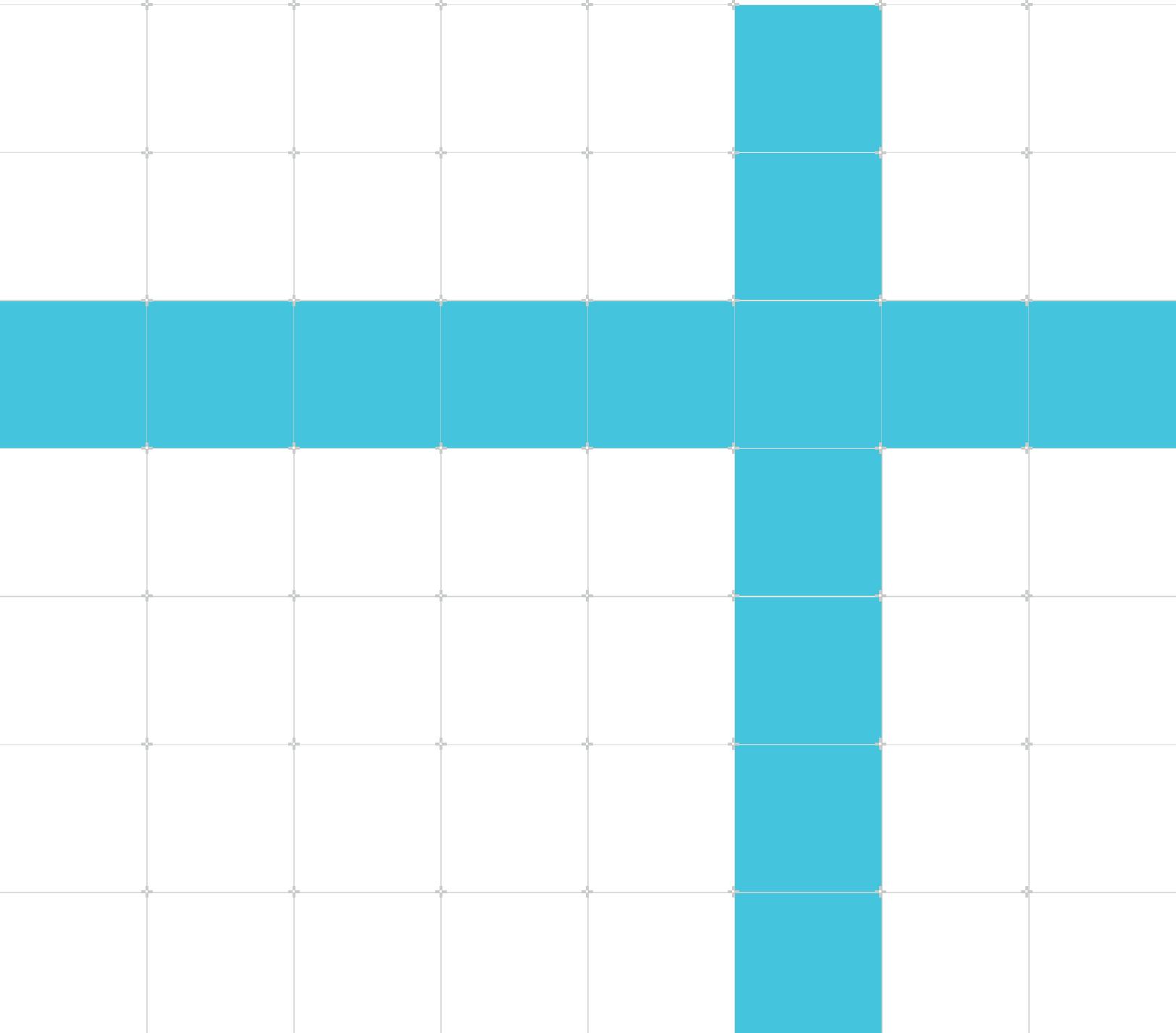
## guide

**Non-Confidential**

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

**Issue 01**

107550\_1.0\_01\_en



## Help with connecting to new targets guide

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

### Release information

#### Document history

Issue	Date	Confidentiality	Change
0100-01	4 August 2022	Non-Confidential	Initial 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, has undertaken no analysis to identify or understand the scope and content of, third party 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 subsidiaries) 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 © 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)

## 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](mailto:terms@arm.com).

# Contents

1. Overview.....	7
2. How do I debug a target with DS-5 or Arm DS?.....	8
3. My target is not listed as a supported platform for DS-5 or Arm DS. How do I debug the target?.....	9
4. I have been given a configuration database or a platform configuration. How do I use it?.....	12
5. What debug probes or connection types can I use to autodetect and connect to my target?...	13
6. I need help connecting to my target. Where do I go?.....	14
7. I am working with a certain target. Are there existing resources to help me get started?.....	15
8. Platform Configuration Editor (PCE) autodetection did not work. Why?.....	16
9. Locked devices on the scan chain.....	19
10. Secure devices on the target.....	20
11. Debug Access Port (DAP) or other components on the JTAG scan chain are not powered up.....	21
12. Target is running too slowly.....	22
13. PCE is receiving unrecognized device IDs.....	23
14. Target not set up for JTAG communication.....	24
15. Need Serial Wire Debug (SWD) rather than JTAG.....	25
16. Reset signal strength used by the debug probe is not correct for the target.....	26
17. Debug connector or signal issues.....	27
18. Debug probe or debug cable issues.....	28

19. DAP cannot be powered up.....	29
20. Wrong number or type of Access Ports (APs) found.....	30
21. AHB-AP or APB-AP ROM Table cannot be found.....	31
22. AHB-AP or APB-AP ROM Table is incomplete or incorrect.....	32
23. PCE could not get the identifying information for the devices behind the DAP.....	33
24. I see unexpected messages in the PCE Console. What do these messages mean?.....	34
25. I see unexpected messages at the top of the SDF file. What do these messages mean?.....	36
26. When my target is autodetected, why do some component or component connections not appear?.....	37
27. Unrecognized device IDs.....	39
28. Cannot find AHB-AP or APB-AP ROM Table.....	40
29. Incorrect or incomplete AHB-AP or APB-AP ROM Table.....	41
30. No identifying information for the devices behind the DAP.....	42
31. Invisible component connections.....	43
32. Component connection is not visible through PCE testing.....	44
33. What resources are there to customize my platform configuration or connection options?...	45

# 1. Overview

Learn about common questions and answers when connecting to a new target with Arm Development Studio (Arm DS) and Arm DS-5 (DS-5). This page focuses on autodetection, connection, and debug with new or custom emulation, FPGA, and silicon targets.

If you cannot find solutions to your questions here:

- Click [I need help connecting to my target. Where do I go?](#)
- Visit the [Software Tools Community](#) where you can ask questions of the Arm experts.

## General new target knowledge

- [How do I debug a target with DS-5 or Arm DS?](#)
- [My target is not listed as a supported platform for DS-5 or Arm DS. How do I debug the target?](#)
- [I have been given a configuration database or a platform configuration. How do I use it?](#)
- [What debug probes or connection types can I use to autodetect and connect to my target?](#)

## New target help

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

## Platform Configuration Editor autodetection

- [Platform Configuration Editor \(PCE\) autodetection did not work. Why?](#)
- [I see unexpected messages in the PCE Console. What do these messages mean?](#)
- [I see unexpected messages at the top of the SDF file. What do these messages mean?](#)
- [When my target is autodetected, why do some component or component connections not appear?](#)

## 2. How do I debug a target with DS-5 or Arm DS?

To debug a target with Arm DS or DS-5, you must have the following:

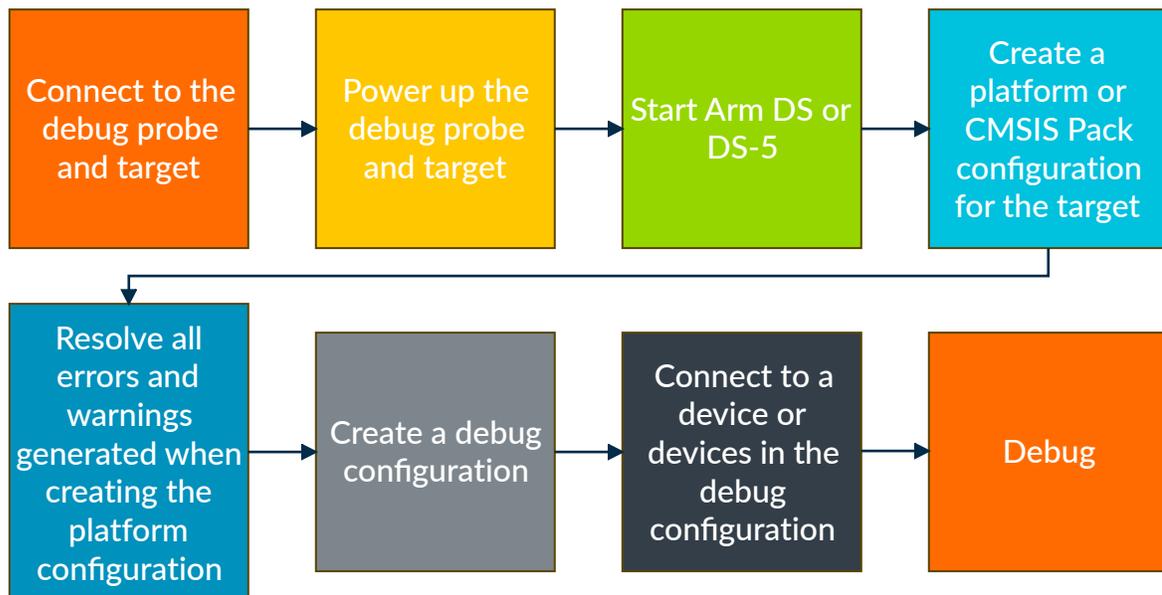
1. A debug probe, like [DSTREAM](#), [DSTREAM-ST](#), [DSTREAM-PT](#), [DSTREAM-HT](#), [DSTREAM-XT](#), and [ULINKpro family](#) debug probes, or a Functional I/O connection.
  - See [What debug probes or connection types can I use to autodetect and connect to my target?](#) for more information.
2. An Arm DS or DS-5 installation that supports the processor you want to debug.
  - Read [Arm DS Compare Editions](#) for information on the different Arm DS editions.
3. An Arm DS or DS-5 license that enables your Arm DS or DS-5 edition.

The following diagram shows the debug flow when using a new target with Arm DS or DS-5:



DS-5 does not support CMSIS-Pack configurations.

**Figure 2-1: Debugger flow digram**



### 3. My target is not listed as a supported platform for DS-5 or Arm DS. How do I debug the target?

Arm Development Studio (Arm DS) and Arm DS-5 Development Studio (DS-5) connects to and debugs targets using a platform configuration in a configuration database (configDB). In Arm Development Studio, connecting is also possible using a CMSIS Pack. Your tools license determines which processors you can debug.

The following section lists debugging information for Arm DS and DS-5:

#### If you are using Arm Development Studio

Arm Development Studio (Arm DS) uses two methods to connect to hardware targets: configuration database (configDB) and CMSIS Packs. Read [What are CMSIS software components?](#) for an overview of CMSIS.

Arm Development Studio manages CMSIS Packs using the CMSIS-Pack Manager perspective. Watch [Introduction to Pack Manager](#) for an introduction to the Development Studio CMSIS-Pack Manager.



Your Arm Development Studio license determines which processors you can debug. [Arm DS Compare Editions](#) contains a comparison table which lists the processors each Arm Development Studio edition supports.

---

The Arm Development Studio has a Hardware Connection wizard to create hardware connections. The Hardware Connection wizard lists the configDB and CMSIS Packs Arm Development Studio supports. [Arm DS Supported device list](#) lists which configDB platforms Arm Development Studio supports. [CMSIS Packs](#) lists the CMSIS-Packs available.

If your target is not listed, use the Hardware Connection wizard to create a platform configuration. The Hardware Connection wizard uses the Platform Configuration Editor (PCE) to generate platform configurations for targets. Generate a platform configuration with Arm Development Studio using:

- [Platform Configuration section of the Arm Development Studio User Guide](#) Documentation
  - Arm Development Studio documentation about creating platform configurations with Arm Development Studio.
- [New Hardware Connection](#)
  - A tutorial video which shows you the steps when connecting Arm Debugger to any hardware target.
- [How to configure debug and trace](#)

- A KBA describing the PCE autodetection process and common PCE Console view messages.

Further tutorials on PCE are available at:

Some of the following resources focus on the DS-5 tools approach, but much of the content is still relevant for Arm Development Studio.

- [SoC Bring-Up Using the Platform Configuration Editor PCE](#)
  - A tutorial showing how to use PCE to detect the underlying system architecture of a SoC and configure any missing system components.
- [PCE simple platform configuration using a DSTREAM target connection](#)
  - A video on how to use PCE to automatically detect and add targets to the configuration database of the debugger.
- [In depth analysis of autoconfiguring with PCE](#)
  - A tutorial showing the different functionality of the Platform Configuration Editor (PCE).
- [How PCE identifies the CoreSight components on the target board](#)
  - A tutorial describing the PCE process and common PCE issues.

When connecting to a new target, it is recommended that users try autodetecting the target with PCE first. If autodetection does not complete, read the [Platform Configuration Editor \(PCE\) autodetection did not work. Why?](#) section of this page.

### If you are using DS-5

Look up the platforms DS-5 supports. If your target is not listed on this page, create a platform configuration for your target.

If your processor is not listed in DS-5, try connecting and debugging the processor with DS-5 anyway. Debugging might work because DS-5 provides generic Arm architecture support. If you are still unable to connect to or debug the processor, refer to the [I need help connecting to my target. Where do I go?](#) section of this page.

Note: Your DS-5 license determines which processors you can debug.



DS-5 is no longer being developed, so no further processor or platform support is available.

---

Eclipse for DS-5 uses a perspective called DS-5 Configuration to add platform configurations to DS-5. The DS-5 Configuration perspective uses the Platform Configuration Editor (PCE) to generate platform configurations.

Generate a platform configuration with DS-5 using:

- [SoC Bring-Up Using the Platform Configuration Editor PCE](#)

- A tutorial that shows how to use PCE in DS-5 to detect the underlying system architecture of a SoC and configure missing components.
- [PCE simple platform configuration using a DSTREAM target connection](#)
  - A video on how to use PCE to automatically detect and add targets to the configuration database of the debugger.
- [Platform Configuration section of the Arm DS-5 Debugger User Guide](#)
  - DS-5 documentation about creating platform configurations with DS-5.
- [In depth analysis of autoconfiguring with PCE](#)
  - A tutorial showing the different functionality of the Platform Configuration Editor (PCE).

When connecting to a new target, it is recommended that users try autodetecting the target with PCE first. If autodetection does not complete, read the [Platform Configuration Editor \(PCE\) autodetection did not work. Why?](#) section of this page.

## 4. I have been given a configuration database or a platform configuration. How do I use it?

Arm Development Studio and DS-5 use platform configurations to connect to boards. Platform configurations are stored in configuration databases (configDBs). If you are given a platform configuration or a configuration database, add it to the debugger using the following tool-specific steps:

### If you are using Arm Development Studio

[Add a configuration database section of the Arm Development Studio User Guide](#) shows how to add a configuration database to Arm Development Studio.

[Add a Platform Configuration to a Configuration Database section of the Arm Development Studio User Guide](#) shows how to add a platform configuration to Arm Development Studio.

### If you are using DS-5

[Adding a configuration database to DS-5 section of the Arm DS-5 Debugger User Guide](#) shows how to add a configuration database to DS-5.

[Installing Debug Support Packages for Custom or Third Party Devices](#) tutorial shows how to add a platform configuration to DS-5.

## 5. What debug probes or connection types can I use to autodetect and connect to my target?

Choose the tool you are using for further information:

### If you are using Arm Development Studio

Arm Development Studio supports autodetecting boards with the [DSTREAM](#), [DSTREAM-ST](#), [DSTREAM-PT](#), [DSTREAM-HT](#), and [DSTREAM-XT](#), and [ULINKpro family](#) debug probes. From Arm Development Studio 2020.0, target connection using the ST-LINK debug probe from STMicroelectronics is supported.

You can add other debug probes, referred to as third party probes or 3PP, to Arm Development Studio for connection, autodetection, and debug purposes. The [Third-party Debug Probe API section of the Arm Development Studio User Guide](#) and the [Add a third party debug probe section of the Arm Development Studio User Guide](#) explain how to add a non-Arm debug probe to Arm Development Studio. [Easier to add support for 3rd party probes](#) shows how to add support for third party probes.



The preceding video focuses on DS-5, but most of the content remains the same for Arm Development Studio.

---

If your target uses [CoreSight SoC-600](#), which conforms to the [ARM Debug Interface Architecture Specification ADIV6.0](#), you might be able to connect and debug your target using Functional I/O. Some Functional I/O connections types are USB, PCIe, and TCP IP. The [Debug and trace over functional I/O section of the Arm Development Studio User Guide](#) and the [Add a debug connection over functional I/O section of the Arm Development Studio User Guide](#) describe how to use Functional I/O with Arm Development Studio. [Support for ADIV6](#) shows how to use Functional I/O.

### If you are using DS-5

DS-5 supports autodetecting boards with [DSTREAM](#) and [DSTREAM-ST](#). If you are using DS-5 v5.29.x, DS-5 supports autodetecting boards with the [ULINKpro family](#) debug probes.

If you are using DS-5 5.29.x, you can add other debug probes, referred to as third party probes or 3PP, to DS-5. The [Adding a third-party debug probe section of the Arm DS-5 Debugger User Guide](#) describes how to add a custom debug probe to DS-5. [Easier to add support for 3rd party probes](#) shows how to add support for third party probes.

## 6. I need help connecting to my target. Where do I go?

If you need help with connecting to your target with Arm Development Studio or DS-5, read the following:

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

## 7. 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 DS-5 or Arm Development Studio:

- [Arm Development Studio Heterogeneous system debug with Arm DS](#)
  - An Arm DS document on how to set up a NXP i.MX7 SABRE development board and use it to debug a Linux image on Cortex-A cores, and bare-metal applications on Cortex-M cores.
- [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 a NXP Semiconductors Vybrid Controller Tower System module.
- [DS-5 support with the HiKey 960 board](#)
  - A blog on connecting to a Hikey 960 board with DS-5.
- [Getting started with DS-5 and i.MX7](#)
  - A blog on using i.MX7 with DS-5.
- [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.
- [Third Party Targets](#)
  - A page containing information on third party targets supported by Arm Development Studio.

## 8. Platform Configuration Editor (PCE) autodetection did not work. Why?

PCE autodetection might not work for various reasons. The following section describes lists and questions that cover common reasons why autodetection fails and provides steps to resolve the issue.

### No devices or an incorrect number of devices is discovered on the JTAG scan chain

Most external debuggers communicate with the JTAG scan chain of the target either through JTAG or Serial Wire Debug (SWD). PCE interrogates the devices on the JTAG scan chain to determine the number and type of devices on the scan chain. The following lists common reasons why PCE might not find any devices or an incorrect number of devices on the JTAG scan chain:

- [Locked devices on the scan chain](#)
- [Secure devices on the target](#)
- [Debug Access Port \(DAP\) or other components on the JTAG scan chain are not powered up](#)
- [Target is running too slowly](#)
- [PCE is receiving unrecognized device IDs](#)
- [Target not set up for JTAG communication](#)
- [Need Serial Wire Debug \(SWD\) rather than JTAG](#)
- [Reset signal strength used by the debug probe is not correct for the target](#)
- [Debug connector or signal issues](#)
- [Debug probe or debug cable issues](#)

### The JTAG scan chain is complete, but PCE did not find any devices to debug

If you have a target that uses the [CoreSight](#) debug infrastructure, the components to debug are usually behind a Debug Access Port (DAP).

DAPs and possibly other JTAG scan chain components are found when autodetecting a target with PCE. PCE might not find any components to debug.

The following lists common reasons PCE might not find any components to debug behind a DAP:

- [DAP cannot be powered up](#)
- [Wrong number or type of Access Ports \(APs\) found](#)
- [AHB-AP or APB-AP ROM Table cannot be found](#)
- [AHB-AP or APB-AP ROM Table is incomplete or incorrect](#)
- [PCE could not get the identifying information for the devices behind the DAP](#)

The following list provides resources to help you diagnose Platform Configuration Editor (PCE) autodetection issues:

- [How PCE identifies the CoreSight components on the target board](#)
  - A tutorial describing the PCE process and common PCE issues.
- [Interpreting the PCE log following autoconfigure section of the In depth analysis of autoconfiguring with PCE tutorial](#)
  - A tutorial section that explains some of the content found in the autoconfiguration log that PCE produces.
- [What can be done when autoconfigure fails section of the In depth analysis of autoconfiguring with PCE tutorial](#)
  - A tutorial section that explains common reasons why the autoconfigure process can fail.
- [How to configure debug and trace KBA](#)
  - A KBA describing the PCE autodetection process and possible error messages that might appear.
- [Debug over power-down blog](#)
  - General information about processor power down and how it can affect a debugger.
- [Initializing a target section of the Before debugging on Armv8-A Developer Guide](#)
  - Guide section that covers extra steps to get a debugger connection
- [Target state section of the Before debugging on Armv8-A Developer Guide](#)
  - Guide section that covers possible target states on debugger connection.
- [Debugging over powerdown section of the 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.
    - For Arm Development Studio, [Debugging Armv8 platforms with CSAT](#) tutorial
    - For DS-5, [Low Level Debug using CSAT on Armv8-based Platforms](#) tutorial
    - [Low Level Debug using CSAT on Armv7-based Platforms](#) tutorial
- [How to use the DAP logger tool KBA](#)
  - Arm Development Studio and DS-5 ship 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 can I use the ULINK debug probe to collect the debug log in Arm DS? 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.

If you need assistance with bringing up your target, refer to the [I need help connecting to my target. Where do I go?](#) section of this page.

## 9. Locked devices on the scan chain

Some targets have one or more locked devices on the JTAG scan chain. Program these devices to make the JTAG scan chain accessible.

Steps:

- Read [What is lock device?](#).

## 10. Secure devices on the target

Some targets have one or more secure devices on the JTAG scan chain or on the target. Program these devices to make the JTAG scan chain and components on the target accessible for debug.

Steps:

- Read Secure devices on my scanchain. (page currently unavailable)

# 11. Debug Access Port (DAP) or other components on the JTAG scan chain are not powered up

A target can have one or more components directly connected to the JTAG scan chain. If any of these connected components are powered down from a debug perspective, the scan chain is not read correctly.

If working with a [CoreSight](#) target, if the DAP is powered down, none of the devices behind the DAP, like the processors and trace components, are found.

Steps:

- Read the [How does PCE detect the information?](#) section of [How PCE identifies the CoreSight components on the target board](#).

## 12. Target is running too slowly

PCE can have difficulties autodetecting a target which is running at low frequencies. Emulation and simulation targets often run at low frequencies.

Steps:

- Read [Arm Development Studio or DS-5 cannot connect to or autodetect a target system with a very slow JTAG or SWD clock](#).

## 13. PCE is receiving unrecognized device IDs

PCE tries to identify components on the JTAG scan chain by clocking out their identifying (ID) information. If PCE cannot find a match for an ID received, the component is not added to the platform configuration.



PCE is mostly only aware of Arm-implemented JTAG components.

---

Steps:

- Read [How PCE identifies the CoreSight components on the target board.](#)

# 14. Target not set up for JTAG communication

On some targets, steps must be taken to enable JTAG communication.

Steps:

- Consult the target designer, manufacturer, or documentation to determine if extra steps are required to establish JTAG communication to the target.
  - If the target has a working debug example image or setup, try using that for testing and comparison purposes to your own custom environment.

## 15. Need Serial Wire Debug (SWD) rather than JTAG

PCE defaults to using JTAG as the debug connection type. Some boards or target connectors only support SWD.

Steps:

- Read [How do I use a Serial Wire Debug \(SWD\) connection?](#)

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

On some targets, you might need to adjust the debug probe reset signal strength for PCE autodetection to work. If the reset signal strength is incorrect for the target, autodetection does not complete successfully.



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

---

Steps:

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

## 17. Debug connector or signal issues

Sometimes debug connector or the debug signals are not working as expected. Problems with debug connectors or signals can prevent Arm Development Studio or DS-5 from connecting to a target or cause unstable debug connections.

Steps:

- Read [Debug Connector Issues](#).

## 18. 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 [TCP and UDP usage for DSTREAM family units](#).
- Read [I have an Arm DS or DS-5 platform configuration. Why can I not connect to my board?](#)
- If you are using a ST-LINK debug probe from STMicroelectronics, read [Problems connecting to target hardware using Arm Development Studio and the ST-LINK debug probe](#).

## 19. DAP cannot be powered up

If the DAP is powered down, no components are found behind the DAP. If PCE is unable to power up the DAP, in the PCE Console view, a `Failed to power up DAP` error message appears.

Steps:

- Read [Understanding the CoreSight DAP](#).

## 20. Wrong number or type of Access Ports (APs) found

The DAP has APs that allow accesses to devices for debug and trace purposes. The Platform Configuration Editor (PCE) tries to detect which APs are present behind the DAP. If the APs are not tied off according to the integration documentation, PCE is unable to determine where the APs end, PCE assumes the maximum number of APs are present. In the PCE Console view, this behavior is shown as the autodetection process finding many APs.

Steps:

- Read [Understanding the CoreSight DAP](#).

## 21. AHB-AP or APB-AP ROM Table cannot be found

If components are connected to a Debug Port (DP) or a Memory Access Port (MEM-AP), the DP or MEM-AP has a ROM Table. The ROM Table provides a listing of what components are attached to the DP or MEM-AP. If PCE does not find a ROM Table for a DP or MEM-AP, a `No ROM table is present on this AP` message appears in the PCE Console view. If you see this message for a DP or MEM-AP that does have a ROM Table, do the following step.

Steps:

- Read [Understanding the CoreSight DAP](#).

## 22. AHB-AP or APB-AP ROM Table is incomplete or incorrect

There are several ways a ROM Table can be incorrect or incomplete. Incorrect or incomplete ROM Tables cause components on the target not to be added to the platform configuration. The following is a list of common ROM Table issues:

- If the PRESENT bit is not set for a ROM Table entry, the PCE Console view shows the message `Entry present bit not set, no device interrogation will occur`. If the PRESENT bit is not set, PCE ignores the ROM Table entry. The corresponding component is not added to the platform configuration.
- If a ROM Table contains the wrong address for a nested ROM Table, the components listed in the nested ROM Table are not added to the platform configuration.
- A ROM Table is terminated correctly by having a `0x0` entry or having an entry at ROM Table offset `0x0EFC`. If the ROM Table is not terminated correctly, the PCE autodetection process might not find all the components on the target.
- If a ROM Table is missing entries for components connected to the associated DP or AP, PCE does not add the components to the platform configuration.

Steps:

- Read [Understanding the CoreSight DAP](#).

## 23. PCE could not get the identifying information for the devices behind the DAP

PCE collects information about the devices behind the DAP to determine what the devices are. The information is collected by reading specific debug registers of the device. If the debug registers are not accessible, the device is not added to the platform configuration. If the debug registers for a device are not accessible, a `Failed to read x bytes from AP <AP number> @ <component debug register base address>` message appears in the PCE Console view.

Steps:

- Read [How PCE identifies the CoreSight components on the target board](#).

## 24. I see unexpected messages in the PCE Console. What do these messages mean?

Even if the PCE autodetection process completes successfully, you might still see unexpected messages in the PCE Console view. For information on the PCE autodetection process and common PCE Console view messages, read the following:

- [How PCE identifies the CoreSight components on the target board](#)
- [How to configure debug and trace.](#)

To test the responsiveness or presence of certain components, you can interact with them using the CoreSight Access Tool (CSAT) or the CoreSight Access Tool for SoC600 (CSAT600). The following lists reference material for CSAT and CSAT600:

- [CoreSight Access Tool \(CSAT\) User Guide](#)
  - The user guide for the CSAT tool.
- [Coresight Access Tool for SoC600 \(CSAT600\) User Guide](#)
  - The user guide for the CSAT600 tool.
- Tutorials that give an overview of performing low-level debug using the CoreSight Access Tool (CSAT) with an Armv8 target:
  - For Arm Development Studio, [Debugging Armv8 platforms with CSAT](#) tutorial.
  - For DS-5, [Low Level Debug using CSAT on Armv8-based Platforms](#) tutorial.
- [Low Level Debug using CSAT on Armv7-based Platforms](#) tutorial
  - A tutorial that provides information about performing some basic debug operations on an Armv7-based platform using the CoreSight Access Tool (CSAT).



The tutorial references DS-5 exclusively, but the content still applies to Arm Development Studio.

---

If you are using a CoreSight target, you can obtain a log of the low-level debug activity. If you are using a [DSTREAM family](#) unit, consult the [How to use the DAP logger tool](#) KBA for instructions on how to capture a DAP log. If you are using a [ULINKpro family](#) unit, consult the [How can I use the ULINK debug probe to collect the debug log in Arm DS?](#) KBA for instructions on how to capture a debug log.

---

You must have a good understanding of the following to interpret the logs:



Note

- CoreSight Architecture
  - CoreSight components
  - The target CoreSight topology details
  - Processor-specific debug information, like the CoreSight register offsets
-

## 25. I see unexpected messages at the top of the SDF file. What do these messages mean?

Even if the Platform Configuration Editor (PCE) autodetection process completes successfully, you might still see unexpected messages. These messages appear:

- In the Summary dialog shown at the end of the autodetection process
- At the top of the platform configuration SDF file, .sdf file extension

If the messages are in yellow text, the messages are warnings. If the messages are in red text, the messages are errors.

Steps:

- Read [Common .sdf errors and warnings and what can be done to solve them](#).

## 26. When my target is autodetected, why do some component or component connections not appear?

For a summary of common reasons why PCE does not discover component or component connections, read the following:

[Common reasons why components and component connections do not appear.](#)

The following list provides common reasons why PCE does not discover components or component connections:

- [Unrecognized device IDs](#)
- [Cannot find AHB-AP or APB-AP ROM Table](#)
- [Incorrect or incomplete AHB-AP or APB-AP ROM Table](#)
- [No identifying information for the devices behind the DAP](#)
- [Invisible component connections](#)
- [Component connection is not visible through PCE testing](#)

The following list offers further help for diagnosing missing component or component connections:

- [How PCE identifies the CoreSight components on the target board](#)
  - A tutorial describing the PCE process and common PCE issues.
- [Creating an extension configuration database in Arm Development Studio and DS-5](#)
  - A KBA on how to create an extension configuration database in Arm Development Studio and DS-5.
- [Debug over power-down](#) blog
  - General information about processor power down and how it can affect a debugger.
- [Initializing a target section of the Before debugging on Armv8-A Developer Guide](#)
  - Guide section that covers extra steps to get a debugger connection
- [Target state section of the Before debugging on Armv8-A Developer Guide](#)
  - Guide section that covers possible target states on debugger connection.
- [Debugging over powerdown section of the 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.
  - For Arm Development Studio, [Debugging Armv8 platforms with CSAT](#) tutorial
  - For DS-5, [Low Level Debug using CSAT on Armv8-based Platforms](#) tutorial
  - [Low Level Debug using CSAT on Armv7-based Platforms](#) tutorial
- [How to use the DAP logger tool](#) KBA
  - Arm Development Studio and DS-5 ship 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 can I use the ULINK debug probe to collect the debug log in Arm DS?](#) 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.

## 27. Unrecognized device IDs

PCE tries to identify components on the JTAG scan chain by clocking out their identifying (ID) information. If PCE cannot find a match for an ID received, the component is not added to the platform configuration.



PCE is mostly only aware of Arm-implemented JTAG components.

---

Steps:

- Read [How PCE identifies the CoreSight components on the target board.](#)

## 28. Cannot find AHB-AP or APB-AP ROM Table

If components are connected to a Debug Port (DP) or a Memory Access Port (MEM-AP), the DP or MEM-AP has a ROM Table. The ROM Table provides a listing of what components are attached to the DP or MEM-AP. If PCE does not find a ROM Table for a DP or MEM-AP, a `No ROM table is present on this AP` message appears in the PCE Console view. If you see this message for a DP or MEM-AP that does have a ROM Table, do the following step.

Steps:

- Read [Understanding the CoreSight DAP](#).

## 29. Incorrect or incomplete AHB-AP or APB-AP ROM Table

There are several ways a ROM Table can be incorrect or incomplete. Incorrect or incomplete ROM Tables cause components on the target not to be added to the platform configuration. The following is a list of common ROM Table issues:

- If the PRESENT bit is not set for a ROM Table entry, the PCE Console view shows the message `Entry present bit not set, no device interrogation will occur`. If the PRESENT bit is not set, PCE ignores the ROM Table entry. The corresponding component is not added to the platform configuration.
- If a ROM Table contains the wrong address for a nested ROM Table, the components listed in the nested ROM Table are not added to the platform configuration.
- A ROM Table is terminated correctly by having a `0x0` entry or having an entry at ROM Table offset `0x0FFC`. If the ROM Table is not terminated correctly, the PCE autodetection process might not find all the components on the target.
- If a ROM Table is missing entries for components connected to the associated DP or AP, PCE does not add the components to the platform configuration.

Steps:

- Read [Understanding the CoreSight DAP](#).

## 30. No identifying information for the devices behind the DAP

PCE collects information about the devices behind the DAP to determine what the devices are. The information is collected by reading specific device debug registers. If the device debug registers are not accessible, the device is not added to the platform configuration. If the debug registers for a device are not accessible, a `Failed to read x bytes from AP <AP number> @ <component debug register base address>` message appears in the PCE Console view.

Steps:

- Read [How PCE identifies the CoreSight components on the target board](#).

## 31. Invisible component connections

Some targets might contain components which have no programmer's model or are not listed in a ROM Table. If either of the preceding is true, the component is invisible to the PCE autodetection process. The component connections might not be added to the generated .sdf file. A `Device x` has multiple connections to master interface `y`, check for correctness warning message might appear at the top of the .sdf file.



This content only applies if you are using an Arm Development Studio version earlier than 2020.0.

---

Steps:

- Read [Common .sdf errors and warnings and what can be done to solve them.](#)

## 32. Component connection is not visible through PCE testing

The Platform Configuration Editor (PCE) interrogates a device debug registers 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. Read [Common reasons why components and component connections do not appear](#) and [How PCE identifies the CoreSight components on the target board](#) for help with determining why the component connection is not found.
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 debug section of the Arm Debugger Manual Configuration Tutorial](#).

## 33. What resources are there to customize my platform configuration or connection options?

The following are some resources available to help customize your platform configuration or connection options in DS-5 or Arm Development Studio:

- [How to Create an ETR Configuration Tab in the DS-5 DTSL Options Dialog](#)
  - A tutorial on how to create and customize an Embedded Trace Router (ETR) tab in the DTSL options dialog in DS-5 using a simple Python script.
- [Creating an extension configuration database in Arm Development Studio and DS-5](#)
  - A KBA on how to create an extension configuration database in Arm Development Studio and DS-5.
- [Add a custom components.xml file to an Arm DS Configuration Database](#)
  - A KBA on how to add a new component definition to the Arm Development Studio a configuration database.