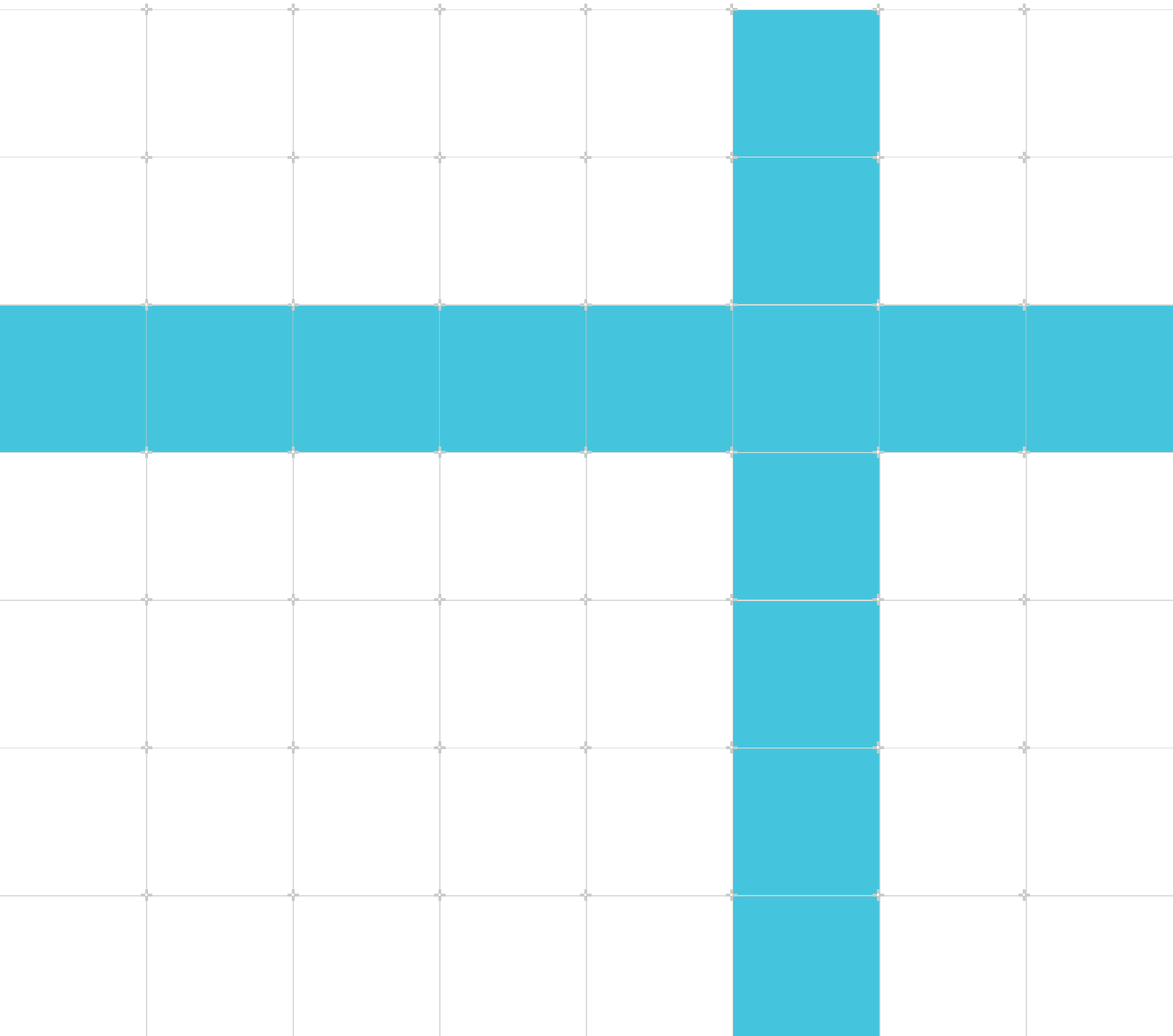
Help with connecting to new targets

Version 2.1

guide

Issue 02
107550_2.1_02_en



Help with connecting to new targets **guide**

Release information

Document history

Issue	Date	Confidentiality	Change
0100-01	4 August 2022	Non-Confidential	Initial release
0200-01	15 September 2022	Non-Confidential	Removes references to DS-5 Development Studio
0200-02	28 February 2023	Non-Confidential	Added more resources

Proprietary Notice

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.

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

Contents

1. Overview

Learn about common questions and answers when connecting to a new target with Arm Development Studio (Arm DS). This content 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 with connecting to my target. Where do I go?
- Visit the Software Tools Community where you can ask questions of the Arm experts.

1.1 General new target knowledge

Read the following sections for general knowledge about debugging with Arm Development Studio (Arm DS):

- How do I debug a target with Arm DS?
- My target is not listed as a supported platform for 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?
- Which resources are there to customize my platform configuration or connection options?

1.2 New target help

Read the following sections for help when working with a new target:

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

1.3 Platform Configuration Editor autodetection

Read the following sections for help with Platform Configuration Editor (PCE) 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?

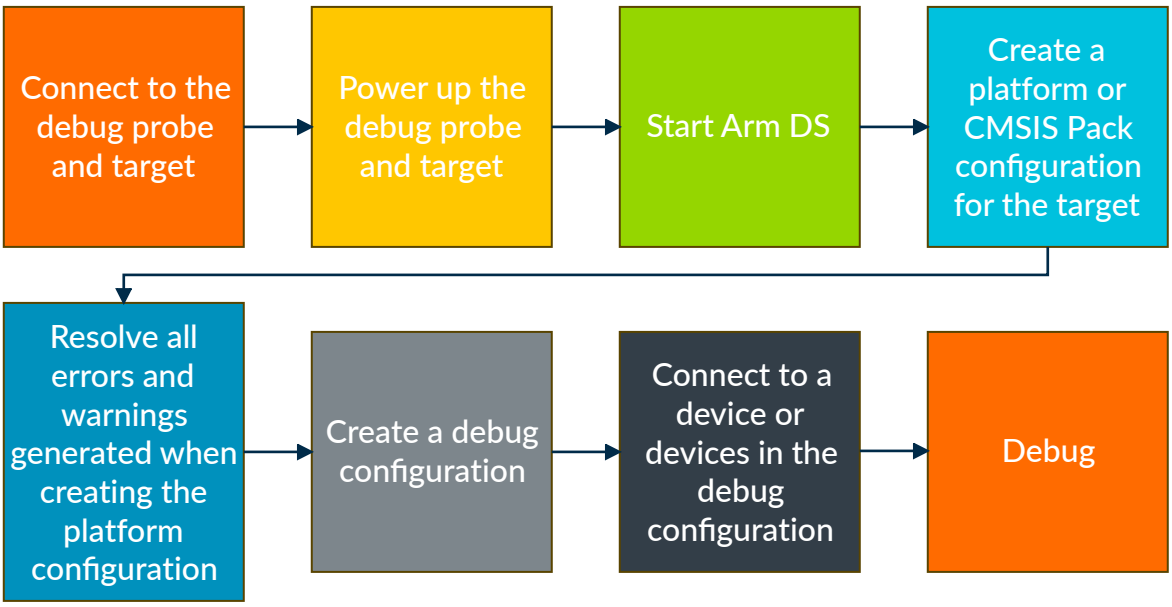
2. How do I debug a target with Arm DS?

To debug a target with Arm DS, 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 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 license that enables your Arm DS edition.

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

Figure 2-1: Debugger flow diagram



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

Arm Development Studio (Arm DS) uses two methods to connect to hardware targets: configuration database (configDB) and CMSIS Packs.

A configDB contains platform configurations. Arm Development Studio uses platform configurations to connect to and debug a target.

What are CMSIS software components? gives an overview of CMSIS. Arm Development Studio manages CMSIS Packs using the CMSIS-Pack Manager perspective. Watch Pack Manager in Development Studio for an introduction to the Development Studio CMSIS-Pack Manager. Read Getting started with Cortex-M using Arm DS and CMSIS for instructions on how to debug a Cortex-M target with CMSIS in Arm Development Studio.

Your Arm Development Studio license determines which processors you can debug. Arm DS Compare Editions contains a comparison table that 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 Devices 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
  - 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 resources on PCE are available at:

- 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 Creation using a DSTREAM Target Connection
    - A video on how to create a PCE simple platform using a DSTREAM target connection
- 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 content.

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

Arm Development Studio uses platform configurations to connect to boards. Platform configurations are stored in configuration databases (configDBs).

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.

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

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 following sections explain how to add a non-Arm debug probe to Arm Development Studio:

- Third-party Debug Probe API section of the Arm Development Studio User Guide
- Add a third party debug probe section of the Arm Development Studio User Guide.

If your target uses CoreSight SoC-600, which conforms to the ARM Debug Interface Architecture Specification ADIv6.0, debugging using Functional I/O might be available. Some Functional I/O connections types are USB, PCIe, and TCP/IP. The following sections describe how to use Functional I/O with Arm Development Studio:

- Debug and trace over functional I/O section of the Arm Development Studio User Guide
- Add a debug connection over functional I/O section of the Arm Development Studio User Guide

Introducing DSTREAM-XT: Debug and trace over functional I/O via PCIe introduces DSTREAM-XT and how it uses Funtional I/O to debug and trace targets.

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

If you need help with connecting to your target with Arm Development Studio, 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 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.

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

PCE autodetection might not work for various reasons. The following sections list common reasons why autodetection fails and provides steps to resolve the issue.

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
  - A KBA describing the PCE autodetection process and possible error messages that might appear.
- Debug over power-down
  - 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. The following are useful resources for CSAT:
    - Debugging Armv8 platforms with CSAT
    - Low Level Debug using CSAT on Armv7-based Platforms

- How to use the DAP logger tool
    - 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?
    - 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 with connecting to my target. Where do I go? section of this page.

8.1 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) not accessible
- 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

8.2 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

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 Securing the debug interface of your devices.

11. Debug Access Port (DAP) not accessible

Some targets have a TAP Controller that must be programmed to access the DAP. Not programming the TAP controller causes the DAP not to be visible when the target is powered on.

Steps:

- Read How do I add pre-connect JTAG scans in Arm DS to enable target connection?

12. 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.

13. 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.

14. 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.

15. Target not set up for JTAG communication

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

Steps:

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

16. Need Serial Wire Debug (SWD) rather than JTAG

PCE defaults to using JTAG as the debug connection type. Some boards only support Serial Wire Debug (SWD). Some boards support both JTAG and SWD, but you want to use SWD for debugging. In either scenario, you must change the debug probe connection mode to use SWD.

Steps:

- Read How do I use a Serial Wire Debug (SWD) connection?

17. 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.

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.

18. 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 from connecting to a target or cause unstable debug connections.

Steps:

- Read Debug Connector Issues.

19. 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?
- To diagnose issues using the LEDs on the top of the DSTREAM-ST debug probe, read the Indictor LEDS on the top section of the DSTREAM-ST 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.

20. 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.

21. 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 how many APs are implemented. If the number of APs implemented cannot be determined, PCE assumes the maximum number of APs are present. In the PCE Console view, this behavior is shown as the autodetection process finding many more APs than expected.

Steps:

- Read Understanding the CoreSight DAP.

22. 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.

23. 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 nested ROM Table address is wrong, 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 `0xEFC`. 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 Debug Port (DP) or Access Port (AP), PCE does not add the components to the platform configuration.

Steps:

- Read Understanding the CoreSight DAP.

24. 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.

25. 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 one of the following tools:

- For ARM Debug Interface Architecture Specification ADIv5.0 to ADIv5.2 or earlier compliant boards, CoreSight Access Tool (CSAT).
- For ARM Debug Interface Architecture Specification ADIv6.0 compliant boards, 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.
- Debugging Armv8 platforms with CSAT
  - A tutorial that provides information about performing low-level debug using the CoreSight Access Tool (CSAT) with an Armv8 target.
- Low Level Debug using CSAT on Armv7-based Platforms
  - A tutorial that provides information about performing low-level debug using the CoreSight Access Tool (CSAT) with an Armv7 target.

If you are using a CoreSight target, you can obtain a log of the low-level debug activity using the following resources:

- If you are using a DSTREAM family unit, read How to use the DAP logger tool. This KBA provides instructions on how to capture a DAP log.
- If you are using a ULINKpro family unit, read How do I get a DAP log when using ULINK family, third-party, or low cost debug probes?. This KBA provides instructions on how to capture a debug log.


Note

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

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

26. 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.

27. 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.
- Debug over power-down
  - 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 states or the target 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)

    - 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. The following are useful resources for CSAT:
        - Debugging Armv8 platforms with CSAT
        - Low Level Debug using CSAT on Armv7-based Platforms
- How to use the DAP logger tool
    - Arm Development Studio 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?
    - 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.

28. 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.

29. 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.

30. 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 nested ROM Table address is wrong, the components 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 `0xEFC`. 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 Debug Port (DP) or Access Port (AP), PCE does not add the components to the platform configuration.

Steps:

- Read Understanding the CoreSight DAP.

31. 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.

32. Invisible component connections

Some targets might contain components that have no programmer 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. If the component is invisible to PCE, 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 at the top of the .sdf file might indicate there is an invisible component.

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.

33. Component connection is not visible through PCE testing

The Platform Configuration Editor (PCE) interrogates the Debug registers of a device 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.

34. Which 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 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 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.
- Add a custom components.xml file to an Arm DS Configuration Database
  - A KBA on how to add component definitions to the configuration database of Arm Development Studio.