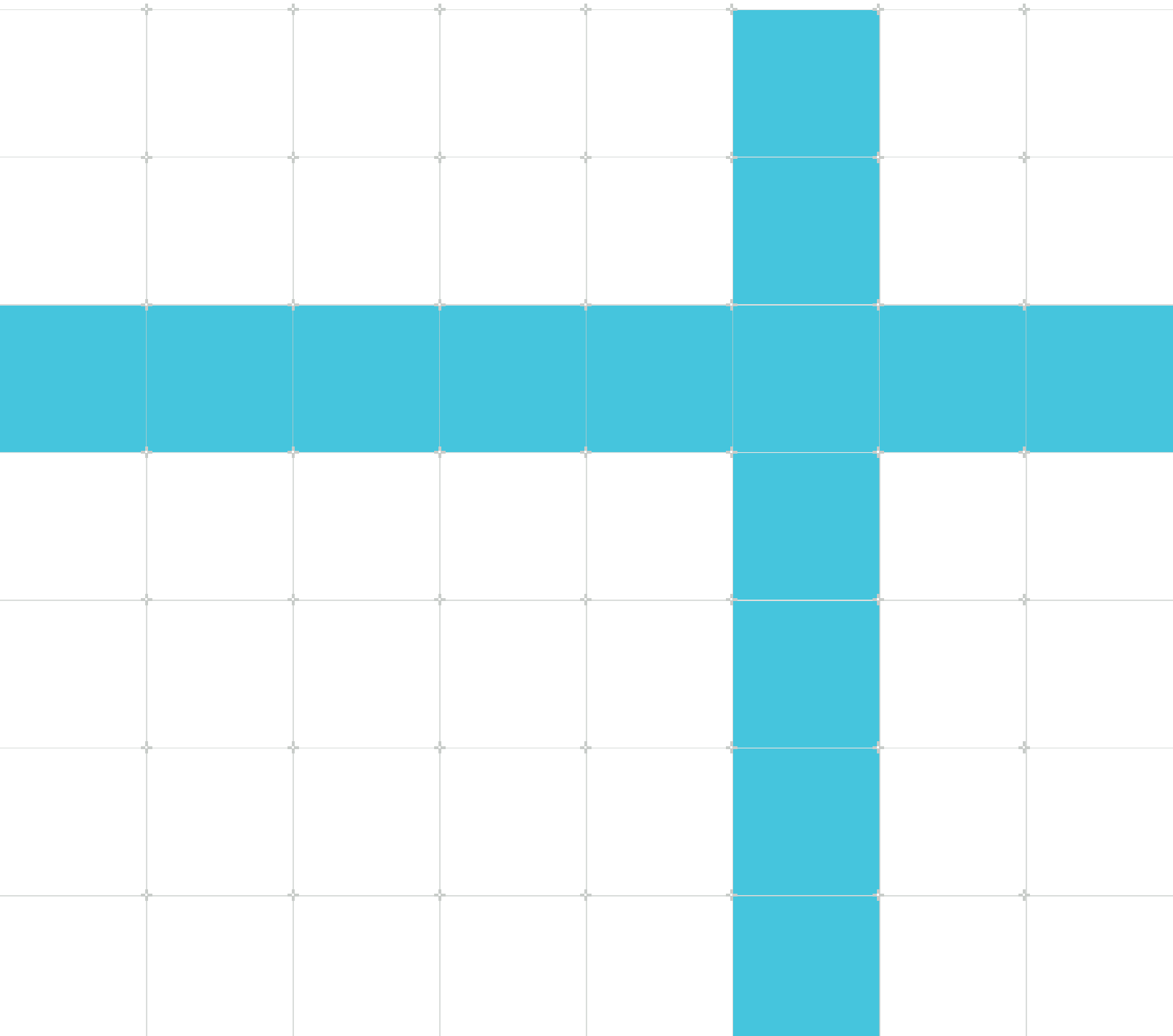
arm

Securing the debug interface of your devices

Version 1.0

Issue 01
107745_0100_01_en



Securing the debug interface of your devices

Release information

Document history

Issue	Date	Confidentiality	Change
0100-01	14 September 2022	Non-Confidential	First release

Proprietary Notice

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

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

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

This document may include technical inaccuracies or typographical errors.

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

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly

or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

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

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

To debug software on a System on Chip (SoC) system like Cortex-M-based microcontrollers, the debugger needs low-level access to the SoCs hardware. This can be a security risk under the right circumstances, as it allows a software developer, which can also be a hacker to access System registers, memories. This includes Secure and normal memories if TrustZone security technology is implemented, and sometimes, the program codes stored in the device. Without proper protection in place for the debug interface, third parties can potentially perform various attacks like:

- Copy firmware and steal valuable software assess for example, algorithms
- Access to cryptography keys or other security sensitive information stored in the system
- Modify the firmware of the device without permission
- Using a compromised device to attack other systems connected to it

To minimize security risks, many silicon providers integrate security measures in their products to disable unauthorized debug accesses. When such protection is enabled, an authorized software developer must unlock the device first before connecting the debugger to the processor system. For modern embedded systems, the authentication process involves a challenge-response mechanism, which is carried out using a communication channel.

To enable debug authentication, a processor system requires several elements:

- Communication interface: This element is required for the communication of challenge-response process. To reduce the total number of pins required for debug, Arm provide CoreSight SDC-600 Secure Debug Channel components. These components provide a communication channel using existing Joint Test Action Group (JTAG) or Serial Wire Debug (SWD) connection. In silicon products that do not have SDC-600, a dedicated communication interface could be used.
- A processing element to process the authentication: This element can be a standalone processor such as Secure Enclave in advanced SoC products, or can be the embedded processor that runs the applications itself. During debug authentication, the processor runs debug authentication firmware to process the authentication messages. Depending on the security requirement, the processor used in a Secure enclave could be specialized processors that provide anti-tampering and features. Arm SecurCore processors (SC300 and SC000) and Cortex-M35P processors are suitable for such scenarios.
- Debug authentication support on the processors – Arm processors provide debug authentication interface to allow debug and trace operations to be enabled or disabled. The processor or a Secure enclave controls these signals, if such subsystem is implemented. For Arm processors that support TrustZone Security Extension, there are separate debug authentication control signal to allow separated Secure and Non-secure debug or trace permission settings.

To make it hard for the attacker to gain debug access, modern debug authentication solutions use cryptographic based techniques. The cryptographic operations can be accelerated using crypto accelerator if available on the SoC. Instead of using cryptographic based authentication, some legacy systems use simple password-based processes to unlock debug access. This process is considered inadequate in modern IoT era.

The disadvantage of having debug authentication is that software developers must undertake extra steps when connecting their debug tools to the development board. In Arm Development Studio - when Platform Configuration Editor (PCE) runs and detects an **UNKNOWN** device, it is possible that the device is a Secure device. If the board does contain a Secure locked device, consult the board designer, manufacturer, or documentation to learn how to unlock the Secure device. You must add an unlock sequence to the platform configurations `.sdf` file. See the KBA How do I add pre-connect JTAG scans to enable target connection? to learn how to unlock the device.

To allow software developers to handle the debug authentications, typically silicon vendors provide software packages for generating or delivering the authentication tokens. The generation of authentication tokens could be handled as a cloud-based service so that only authorized software developers can use. To make the process easier for software developers, the software used by software developers could:

- Connect to the cloud-based authentication service to generate authentication tokens on demand.
- Interface with debug tools directly to send the authentication tokens to the development board directly through the debug interface.

Development tools might also offer extra features to make debug authentication easier. Please note many application processor systems might provide software-based debug (for example, GDB in a Linux environment) for application development. In such scenarios the application being debugged is sandboxed through the security measure of the execution environment and software developers do not have low-level access to hardware resources. Therefore, debug authentication might not be required.

2. Platform Security Architecture Authenticated Debug Access Control

To enable best practices for Secure debug and enable software developers to deal with debug authentication easily, there is a strong demand for standardization of debug authentication technologies. As a result, the Platform Security Architecture (PSA) initiative in Arm has been working on the standardization of Secure debug technology to enable:

- Consistence debug authentication protocols
- Consistence interface between IDEs from tool vendors and authentication software from silicon vendors
- Choices of modern cryptographic based authentication schemes
- Scalability to different systems, for example: multiple processors
- Reference firmware implementation available as open-source software

The PSA ADAC project is supported by multiple silicon vendors and debug tool vendors. When Authenticated Debug Access Control (ADAC) enabled products reach the market, software developers can utilize Secure debug features with a range of debug tools that have PSA ADAC support. This market appearance is expected to be in the second half of 2022.

3. Related information

Here are some resources related to material in this guide:

Platform Security Architecture (PSA):

- Arm PSA security main page
- PSA Authenticated Debug Access Control Specification
- Reference software

SDC-600 Secure Debug Channel:

- Legacy software example based on CryptoCell API (superseded by PSA ADAC)
  - Code on Github
  - Documentation
- Product page

The following resources are related to this guide:

Arm Development Studio:

- Detecting and unlocking the Arm Secure Debug Channel
- Help with connecting to new targets
- Help with debugging and tracing targets
- How PCE identifies the CoreSight components on the target board
- What is lock device?