

AN12419

Secure JTAG for i.MX RT10xx

Rev. 3 — 18 December 2023

Application note

Document information

Information	Content
Keywords	Secure JTAG, i.MX RT10xx, MCU
Abstract	This document describes how the Secure JTAG on the i.MX RT10xx MCU family can be used.



1 Introduction

This document describes how the Secure JTAG on the i.MX RT10xx MCU family can be used.

The i.MX RT series System JTAG Controller (SJC) provides a possibility to regulate the JTAG access. The three JTAG security modes are available in the i.MX RT series:

- No Debug mode—Maximum security is provided in this mode. All security-sensitive JTAG features are permanently blocked, preventing any debug.
- Secure JTAG mode—High security is provided in this mode. A secret key-based challenge/response authentication mechanism is used for JTAG access
- JTAG Enabled mode—Low security is provided in this mode. It is the default mode of operation for the SJC.

Moreover, you can also fully disable the SJC functionality. For configuration of these JTAG modes, One Time Programmable (OTP) eFuses are used and burned after packaging. The fuse burning process is irreversible. It is impossible to revert the fuse back to the unburned state. To explain, Secure JTAG mode is used in this document. The aim is to allow return/field testing. Authorized reactivation of the JTAG port is allowed in this mode. On the HW side, JTAG signals must be routed out and accessible in the application.

Note: This application note is based on legacy tools and flow, but the description of the secure flow within the chip still applies. We recommend using the latest tools (the [MCUXpresso Secure Provisioning \(SEC\) Tool](#) and [SPSDK](#)) instead of following the steps presented here. For any questions, contact your local support.

2 i.MX RT10xx Secure JTAG support

JTAG access is limited in the Secure JTAG mode by using a challenge/response-based authentication. Any access to the JTAG port is internally checked. Only the devices authorized for debugging (with the right response) can access the JTAG port. Otherwise, JTAG access is denied. The external debugger tools (such as SEGGER J-Link, Lauterbach Trace32, Arm RVDS/DS5) supporting the challenge/response-based authentication mechanism can be used. The secure JTAG mode is typically enabled in the factory manufacturing and not used during the development.

2.1 How to put the chip in Secure JTAG mode

The Secure JTAG feature is only available in the **SJC** mode, and can be selected by the JTAG_MOD input pin (GPIO_AD_B0_08). To enable the SJC mode, the pin must be connected to log.1, which means that the Secure JTAG is unavailable in the CM7 DAP mode.

Table 1. JTAG_MOD pin settings

Signal	Description	Pad	Mode	Direction
JTAG_MOD	SJC mode selection. This pin is sampled at TRST reset to determine two possible modes for the TAP connection configuration.	GPIO_AD_B0_08	ALT0	IO

Note: Due to known HW conflicts on i.MX RT10xx EVKBs, small PCB modifications may be required for JTAG_TDO/TDI signals. Otherwise, the communication in JTAG mode is not possible. Refer to the EVKB schematic for changes needed.

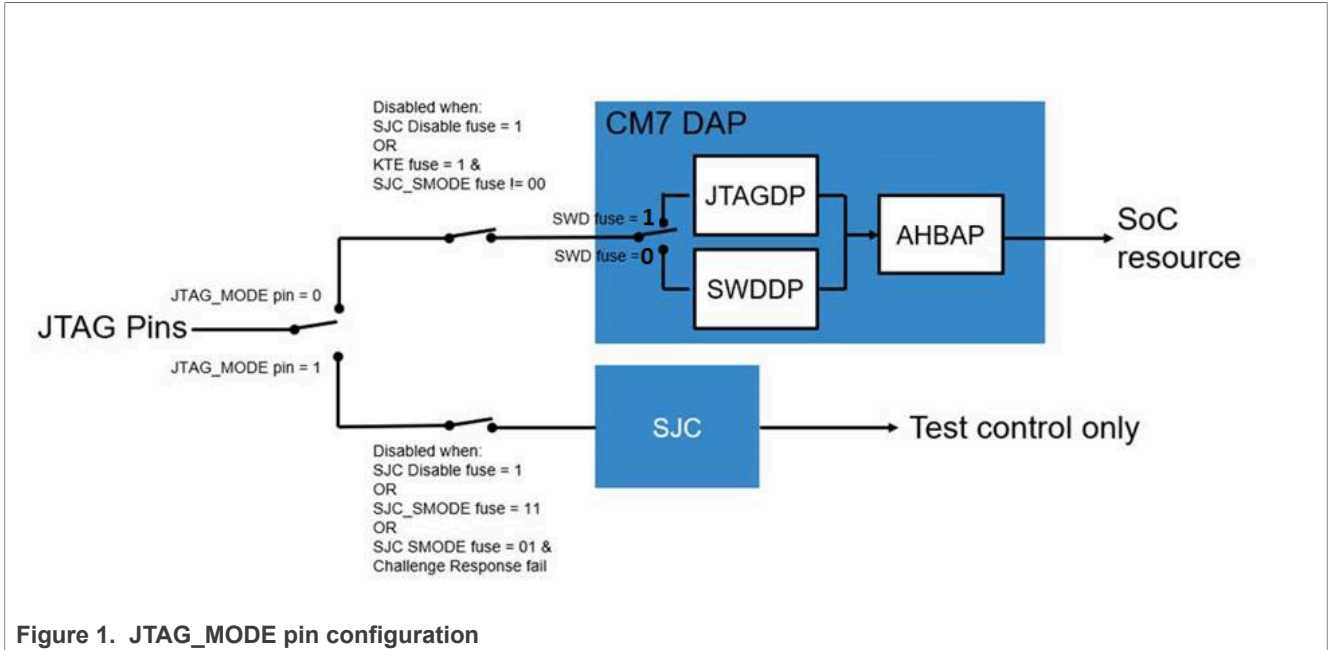


Figure 1. JTAG_MODE pin configuration

2.2 i.MX RT SJC Security Modes

The i.MX RT10xx System JTAG Controller (SJC) supports three different security modes. JTAG enabled is the default mode of operation for the SJC. The user can select the Secure JTAG mode by programming a value 0x1 to the eFuse labeled JTAG_SMODE, described in Table 2. The eFuse has the default value 0x0, which means that the JTAG controller is unsecured by default. Further details on eFuses are available in the Fusemap and On-Chip OTP Controller (OCOTP_CTRL) chapters in the appropriate SRM_RT10xx Security Reference Manual for the i.MX RT1050 available at www.nxp.com upon request.

To lock and prevent further modification to the JTAG_SMODE eFuse, program the BOOT_CFG_LOCK eFuses in addition to programming the JTAG_SMODE eFuse.

Note: Programming these fuses disables access to functions and JTAG Security Mode fuse bits. Ensure that it is programmed last, once the final fuse configuration has been decided. At a minimum, setting the fuse to Override Protect (OP) mode is recommended.

Table 2. eFuses associated with the Secure JTAG feature

Addr[bits]	Fuse Name	Fuse Function	Settings	Locked By
0x460[27]	JTAG_HEO	JTAG HAB Enable Override. Disallows HAB JTAG enabling. The HAB may normally enable JTAG debugging with the HAB_JDE-bit in the OCOTP SCS register. The JTAG_HEO-bit can override this behavior	0 - HAB may enable JTAG debug access 1 - HAB JTAG enable is overridden (HAB may not enable JTAG debug access)	BOOT_CFG_LOCK
0x460[26]	KTE	Kill Trace Enable. Enables tracing	0 - Bus tracing is allowed	BOOT_CFG_LOCK

Table 2. eFuses associated with the Secure JTAG feature...continued

Addr[bits]	Fuse Name	Fuse Function	Settings	Locked By
		capability on ETM, and other modules	1 - Bus tracing is allowed in case the security state as defined by Secure JTAG allows it (for example, JTAG_ENABLE or NO_DEBUG)	
0x460[23:22]	JTAG_SMODE[1:0]	JTAG Security Mode. Controls the security mode of the JTAG debug interface	00 - JTAG enable mode (Default) 01 - Secure JTAG mode 11 - No debug mode	BOOT_CFG_LOCK
0x460[20]	SJC_DISABLE	Additional JTAG mode with the highest level of JTAG protection, thereby overriding the JTAG_SMODE eFuses. In this mode, all JTAG features are disabled, including Secure JTAG and Boundary Scan	0 - JTAG is enabled 1 - JTAG is disabled	BOOT_CFG_LOCK
0x460[19]	DAP_SJC_SWD_SEL	Control DAP works in JTAG or SWD mode	0 - DAP works in SWD mode 1 - DAP works in JTAG mode	BOOT_CFG_LOCK
0x400[3:2]	BOOT_CFG_LOCK[1:0]	Perform lock protection on BOOT-related fuses. This fuse locks numerous functions including JTAG_SMODE	00 - Unlock 1x - Override Protect (OP) x1 - Write Protect (WP) 11 - Both OP and WP	N/A
0x400[6]	SJC_RESP_LOCK	SJC response lock	0 - Unlock 1 - Lock (WP,OP,RP, sense)	
0x600	SJC_RESP[55:0]	Response reference value for the secure JTAG controller		SJC_RESP_LOCK (locks also for read and explicit sense)

Note:

The level of security cannot be reduced but only increased. Since debug modes are controlled by OTP (Hardware fuses), bits can only be blown once.

For example, the following mode changes are possible:

- “JTAG Enabled” to “Secure JTAG”
- “Secure JTAG” to “No debug”

2.3 Secure JTAG eFuses

The challenge/response mechanism used to authenticate the JTAG access uses a challenge value and the associated secret response key. The keys are stored in eFuses inside the IC. The i.MX RT1050 series eFuses used to store the challenge value and the secret response key are listed below:

- The challenge value is the “Device Unique ID” that is programmed into the eFuses. This Device ID is unique for each IC and can be read from the OCOTP registers HW_OCOTP_CFG0 and HW_OCOTP_CFG1. The eFuses are programmed during manufacturing.
- The user program the secret response key (56 bits) into the eFuses marked SJC_RESP.
- The KTE fuse must be programmed to have JTAG secure mode work. Each POR only allows one-time response code input. If the response code is incorrect, the chip must have a POR reset before trying another response code. POR clears sensitive data except the SNVS domain.

After programming the secret response key, the user must disable the ability of software running on the Arm core to read or overwrite the response key. This is done by programming a 0x1 to the associated lock eFuse HW_OCOTP_LOCK_SJC_RESP.

The definition of the response value is left to the user. The Arm core cannot read the value once the response fuse field is provisioned and locked.

2.4 SW Enabled JTAG

The Secure JTAG authentication may be bypassed in SW by writing '1' to the HAB_JDE (HAB JTAG DEBUG ENABLE) bit in the e-fuse controller module. By this JTAG is opened, regardless of its security mode. The S/W JTAG enable allows JTAG enabling without activating the challenge-response mechanism.

The platform initialization software must set the LOCK bit for the JDE bit before transferring control to the application code to ensure that only the trusted SW can set the JDE bit.

The JTAG SW enable does not allow debug in case of boot or memory fault as it requires a reset before entering debug.

The JTAG_JDE bit SW enable backdoor access can be permanently disabled by burning the JTAG_HEO fuse.

Note: *The S/W enabled JTAG feature reduces the overall security level of the system as it relies on S/W protections. If this feature is not required, it is recommended to burn the JTAG_HEO e-fuse that disables this feature.*

2.4.1 JDE bit control in HAB (High Assurance Boot)

The HAB_JDE can be set to '1' by ROM boot SW after unlocking by the Authenticate CSF command.

Before generating of the signed program image, the user must edit the UNLOCK section in the .sb file and provide the device-specific UID in the proper format as a sequence of 8 bytes, see the below example for UID = 0x63e1841b440b81d2:

```
section (SEC_UNLOCK;
Unlock_Engine = "OCOTP",
Unlock_features = "JTAG, SCS, SRK REVOKE",
Unlock_UID = "0xe1, 0x63, 0x1b, 0x84, 0x0b, 0x44, 0xd2, 0x81"
)
```

For more information about the HAB_JDE SW control by platform initialization SW in HAB (High Assurance Boot) refer to section **5.2.13 Unlock (HAB only)** in [HAB Code-Signing Tool User's Guide](#).

2.5 Secure JTAG debug authentication protocol

When the SJC is in Secure Debug mode, the authentication process is as follows:

1. JTAG shifts the challenge key through the Test Data Output (TDO) chain.
2. On the host side, the debug tool takes the challenge key as an input and generates the expected response key.
3. The associated response key is shifted back through the Test Data Input (TDI) chain.
4. The SJC compares the expected internal fused response key with the one shifted in and enables the JTAG access only if it matches.

Note:

Any device reset after JTAG access authorization shifts the JTAG controller back to its locked state.

Figure 2 shows how the challenge/response mechanism works with the JTAG tools.

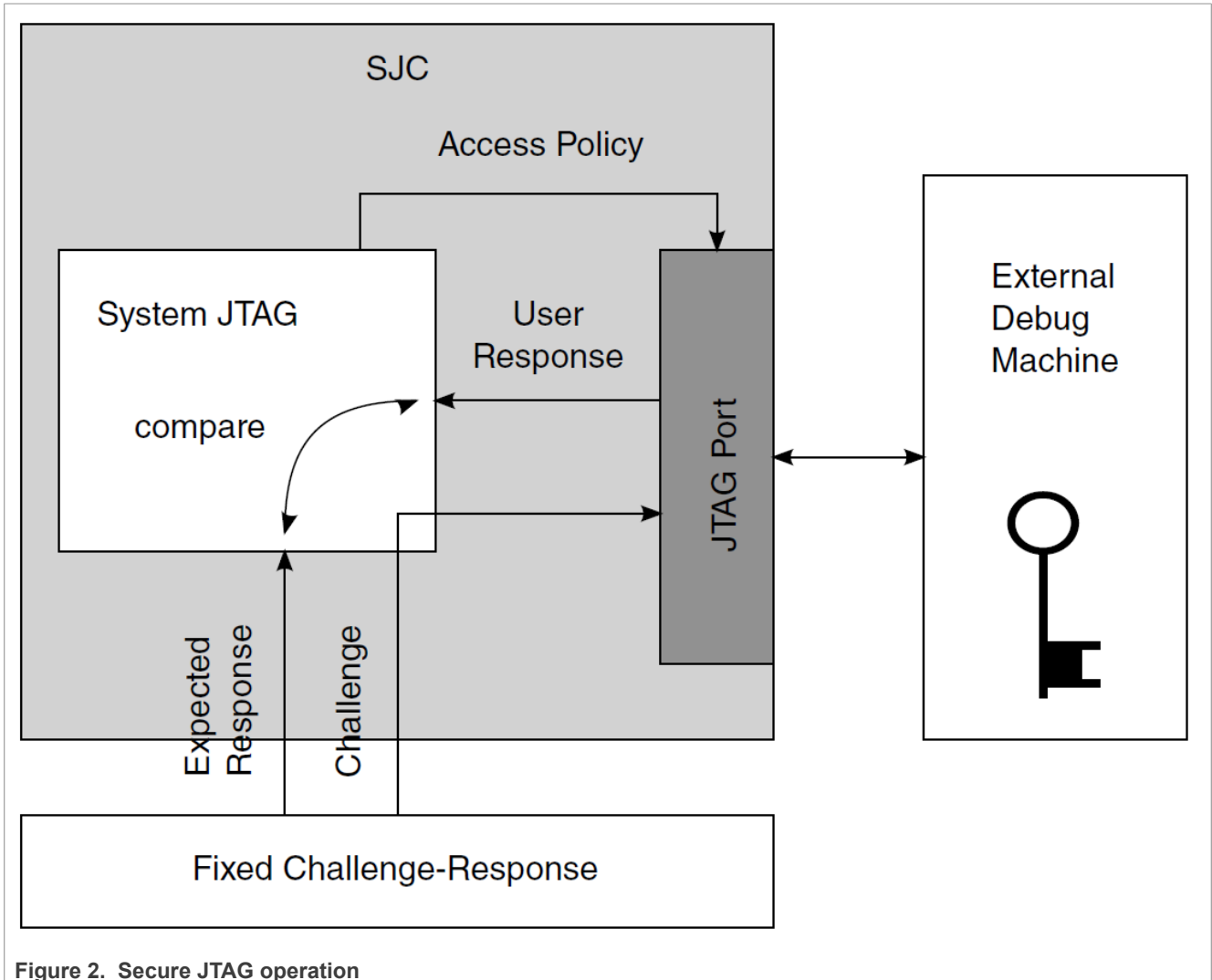


Figure 2. Secure JTAG operation

The JTAG debug tool passes the retrieved challenge key to the user’s application and gets the associated response key in return. The management of the challenge/response pairs is user-dependent and not handled by NXP or the debug tool vendors. Key management is discussed further in [Section 3](#).

2.6 SJC disable fuse

In addition to the various JTAG security modes implemented internally in the SJC, there is an option to disable the SJC functionality with the SJC_DISABLE eFuse. This eFuse creates an additional JTAG mode, JTAG Disabled with the highest level of JTAG protection, overriding the JTAG_SMODE eFuses. In this mode all JTAG features are disabled, including Secure JTAG and Boundary Scan; users must ensure that this fuse is not blown if they wish to use the Secure JTAG functionality.

3 Secret response key approaches

For every challenge value (“Device Unique ID” in i.MX RT10xx) that is retrieved with a JTAG instruction, there is an associated secret response key known only by the user. The JTAG tool vendor only handles the JTAG mechanism used by this authentication process, and does not know the secret response key value programmed into the eFuses. It is left to the user to determine the level of protection that is put in place.

The following are policies for secret response key management by the user application.

1. Identical Response Keys —The same response key is used for each chip. The user can choose a response key that is fused in all chips. This is the simplest, but least sophisticated usage from a security point of view. If an unauthorized user gains access to the fused response key, all the products fused with this response key can be accessed through the JTAG port.
2. Database of Unique Response Keys—The user maintains a database of all generated response keys. The user application can look up the table based on the challenge value. It is possible to implement a secure server holding the challenge/response pairs authenticating the user but this requires an independent implementation effort. The challenge values for all ICs must be read and a database of matching challenge response pairs must be built. Storing and managing numerous response keys is not trivial, but advantageous from a security standpoint, as it does not rely on any breakable algorithms.
3. Algorithmically Generated Response Keys—Response keys are generated based on an algorithm. With this method, there is no large database to manage. For instance, the challenge value can be used by the algorithm to generate a response key. This response key is programmed into SJC_RESP eFuses. Then, every time the challenge value is retrieved through JTAG, it can be processed by the user application and used to generate the expected response key for the JTAG debug tools. Once the algorithm is exposed or reverse-engineered, this method is no longer secure.

Note:

NXP does not provide secure response key management or key generation services; these topics are not within the scope of this document.

3.1 Programming Secure JTAG eFuses using the NXP tool

To program the relevant eFuses needed for Secure JTAG on the chip, follow the steps below. Information on the On-Chip OTP Controller (OCOTP_CTRL) and the Fusemap can be found in the appropriate i.MX RT10xx series reference manual available at www.nxp.com. The NXP MCU Boot Utility is used in the following steps to program eFuses.

1. Download the latest NXP MCU Boot Utility from:<http://www.nxp.com>
2. Program the values below to the eFuses needed for secure JTAG:
 - Read and back-up the 64-bit “Challenge” value stored in the eFuse UUID[1,0], location (0x420, 0x410). See [Figure 3](#).
 - Program a 56-bit (7 Bytes) secret response key in the eFuse SJC_RESP, location (0x610, 0x600). In the example below, value “0xedcba987654321” is programmed.
Note: In [Figure 3](#), MSB is stored on the higher address, while the most-left byte is 0x00 and it is ignored. Define the own response key and keep the key backup for further usage.

- Program 0x1 in the eFuse DAP_SJC_SWD_SEL to switch the DAP to the JTAG mode.
- Program 0x1 in the eFuse JTAG_SMODE to switch the SJC to Secure JTAG mode.
- Program 0x1 in the eFuse KTE_FUSE
- Finally, the user must program 0x1 in the eFuse SJC_RESP_LOCK to disable read/write access of the secret response key. After this operation, the secret response field “SJC_RESP” becomes “invisible” in the fuse map. See [Figure 3](#) and [Figure 4](#).

The following figures demonstrate how to use the NXP MCU Boot Utility to program the above eFuses. The eFuse operation utility is a part of the tool, which can read and write the eFuse map registers. Be careful when doing writing (Burn) operations, as it is irreversible and may lock some features or the device completely.

To have the Secure JTAG enabled, follow the steps mentioned above in [Section 3.1](#) and refer to [Table 2](#), [Figure 3](#) and [Figure 4](#) for more details about the appropriate eFuse bits.

The example does not program the BOOT_CFG_LOCK[1:0] and eFuses to prevent further modifications of the JTAG_SMODE eFuse. As programming these lock eFuses, disables the access to functions in addition to the JTAG mode bits. Therefore, it must be performed once the final configuration has been decided.

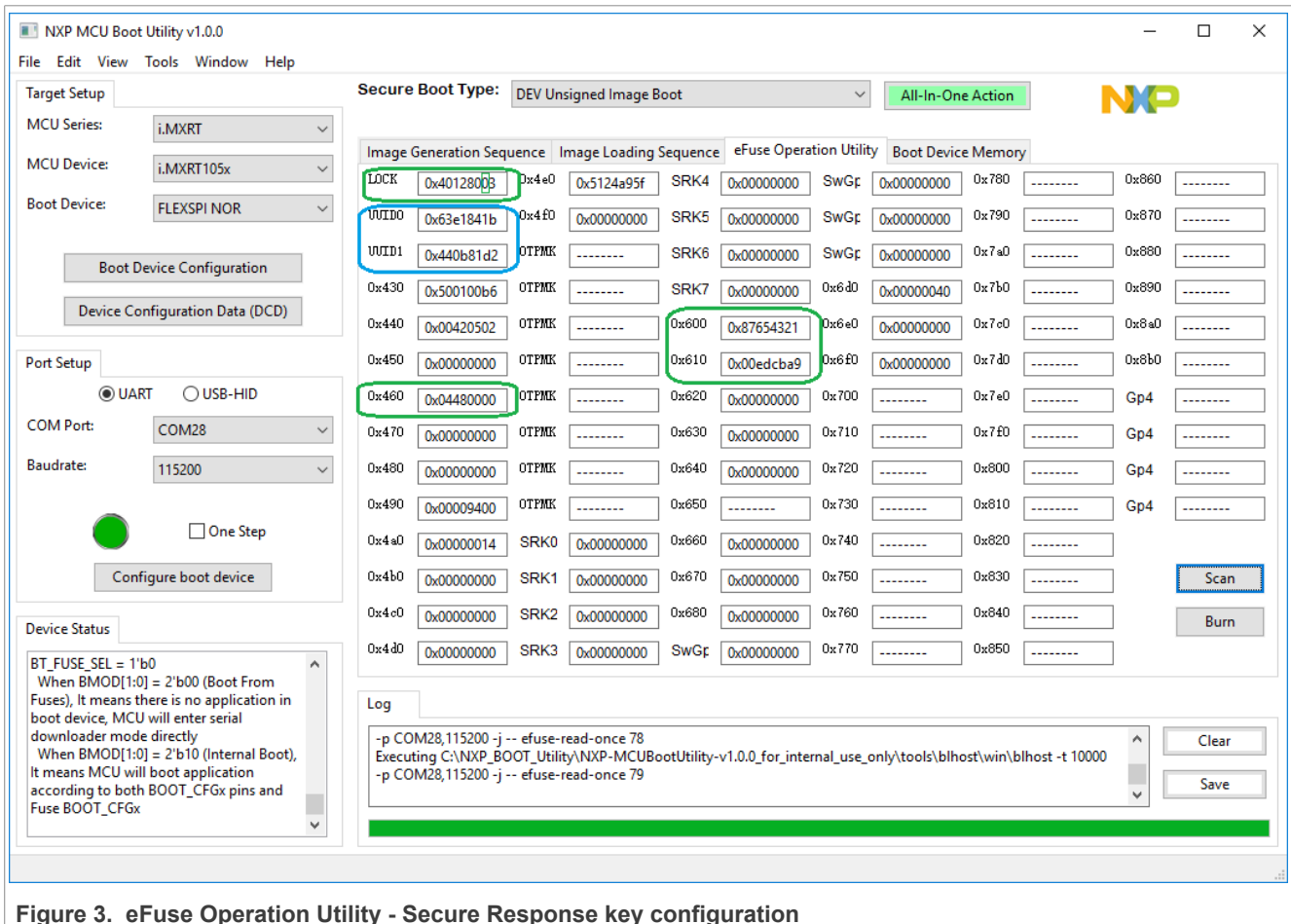


Figure 3. eFuse Operation Utility - Secure Response key configuration

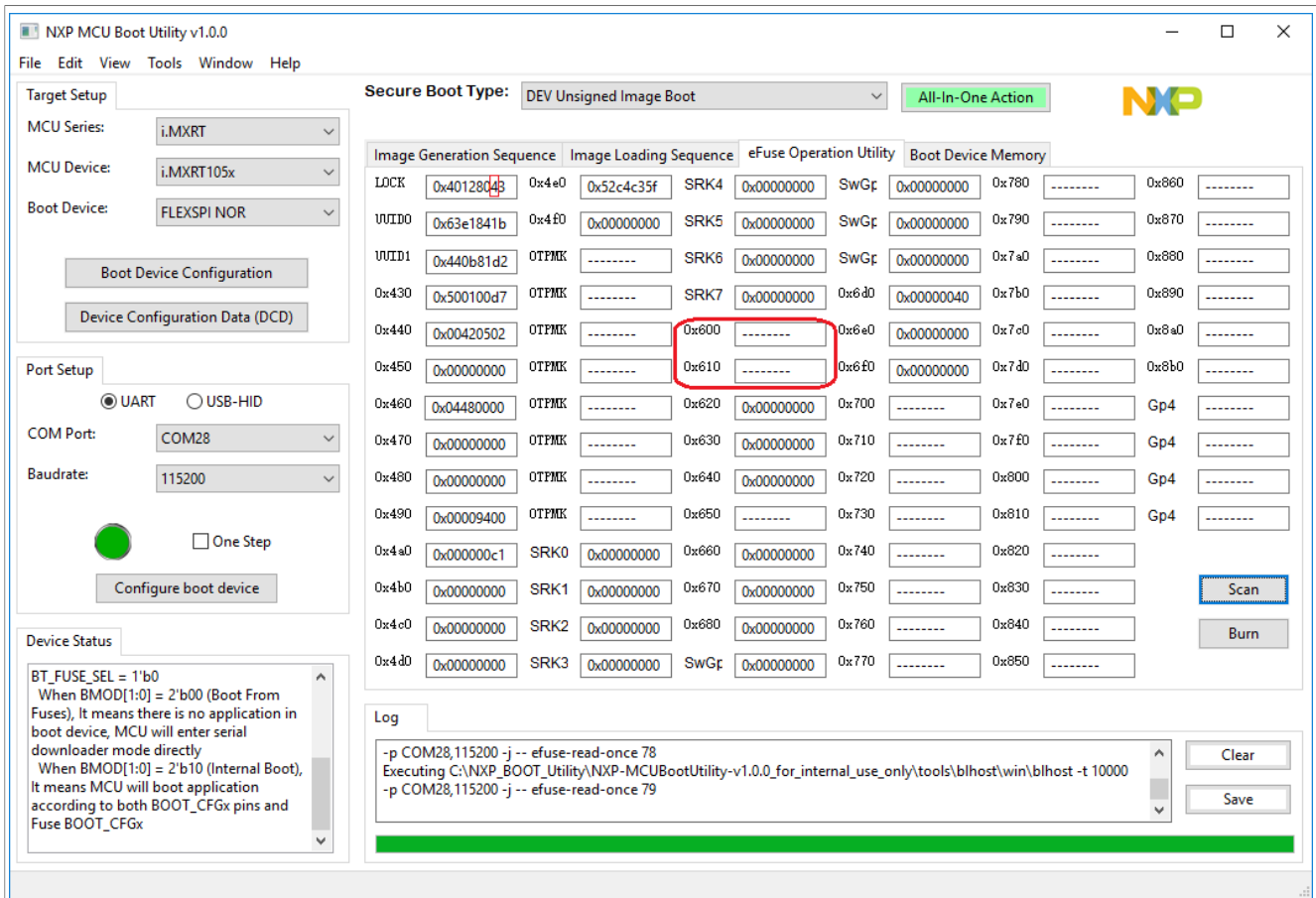


Figure 4. eFuse Operation Utility – Secure Response key lock

4 Debugging with the Secure JTAG enabled

To use the Secure JTAG feature, the JTAG debugger must support it. The example provided in this section uses the SEGGER J-Link debug tool.

Although the procedures outlined in the example below use an i.MX RT1052 device on the IMXRT1050-EVKB board. They can be applied to other RT10xx devices too. The following steps assume that users have experience working with the debug tools and the NXP MCU Boot Utility.

4.1 Steps to connect J-Link debugger via Secure JTAG

The following steps connect the SEGGER J-Link debug tool to the i.MX RT10xx when using Secure JTAG:

1. Download the SEGGER J-Link Software and documentation pack: <https://www.segger.com/downloads/jlink/#J-LinkSoftwareAndDocumentationPack>
If you wish to navigate to these scripts from the SEGGER main page for reference, they are located under “Downloads” - “J-Link / J-Trace” - “J-Link Software and Documentation Pack”.
2. Download and edit the file J-Link script file named “NXP_RT1052_SecureJTAG.JlinkScript”. The script file can be received from NXP upon request. In this file, add the secret response key that was programmed into the SJC_RESP eFuse. In the following example, the secret response key is “Oxedcba987654321”, and matches the response key programmed in the eFuses in [Section 3.1](#) .
// Secure response stored @ 0x600, 0x610 in eFUSE region (OTP memory)

Key0 = 0x87654321;
 Key1 = 0xedcba9;

3. Power-up or reset the board with the JTAG_MODE pin (GPIO_AD_B0_08) in log.1. The user must do it manually, since we do not have the signal connected to the debug connector on EVB.
4. Locate the SEGGER SW J-Link installation directory.
5. Run the “jlink.exe” with the mentioned script file as a parameter.

For instance:

```
jlink.exe -JLinkScriptFile NXP_RT1052_SecureJTAG.JlinkScript -device MCIMXRT1052 -if JTAG -speed 4000 -autoconnect 1 -JTAGConf -1, -1
```

Note: The external IDE tool can call “JLinkGDBServer.exe” application with the same script file to unsecure the target.

The tool script must read the Challenge value from the eFUSE UUID[1,0] location. And it provides the appropriate Response from the SJC for the authentication match.

The JTAG_MODE pin must be switched to log.0, see [Figure 1](#) and [Figure 6](#) for details. Do it manually by changing the pin polarity on the board. If the pin is routed to the JTAG connector and the tool supports the control of this signal, the tool can do it automatically,

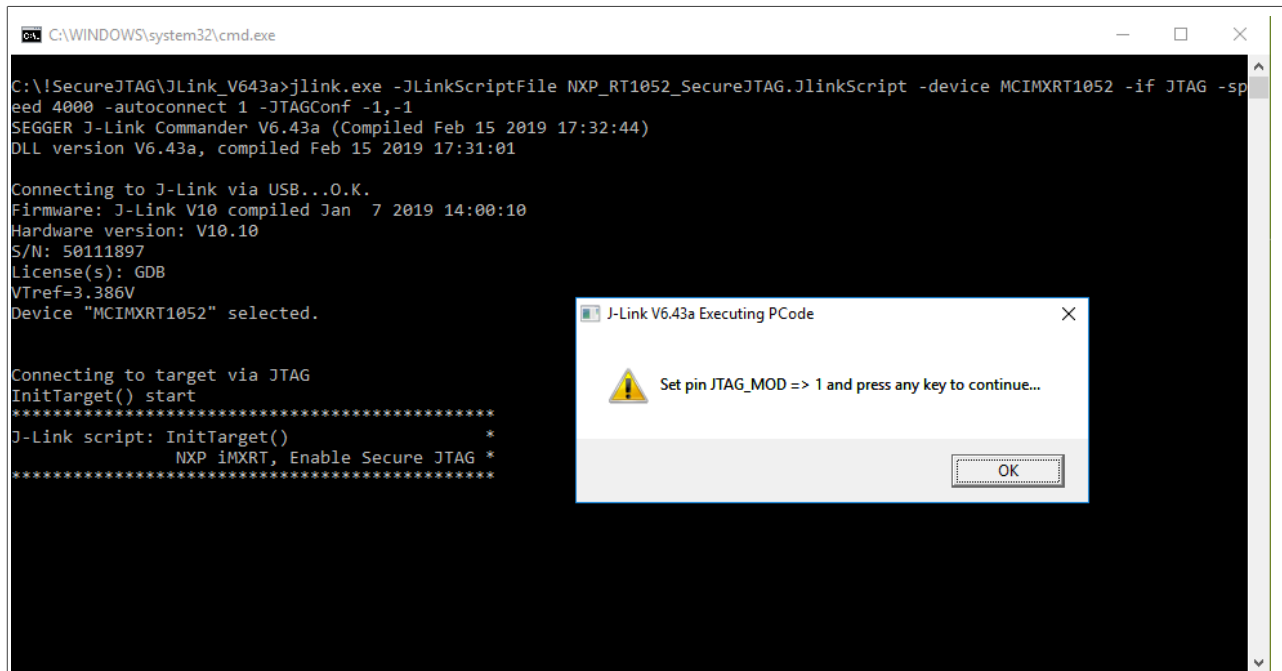


Figure 5. Running SEGGER J-Link script on the secured i.MX RT device

The debug tool should successfully attach to the i.MX RT10xx target over JTAG. The screen capture in [Figure 7](#) shows a successful attach over Secure JTAG:

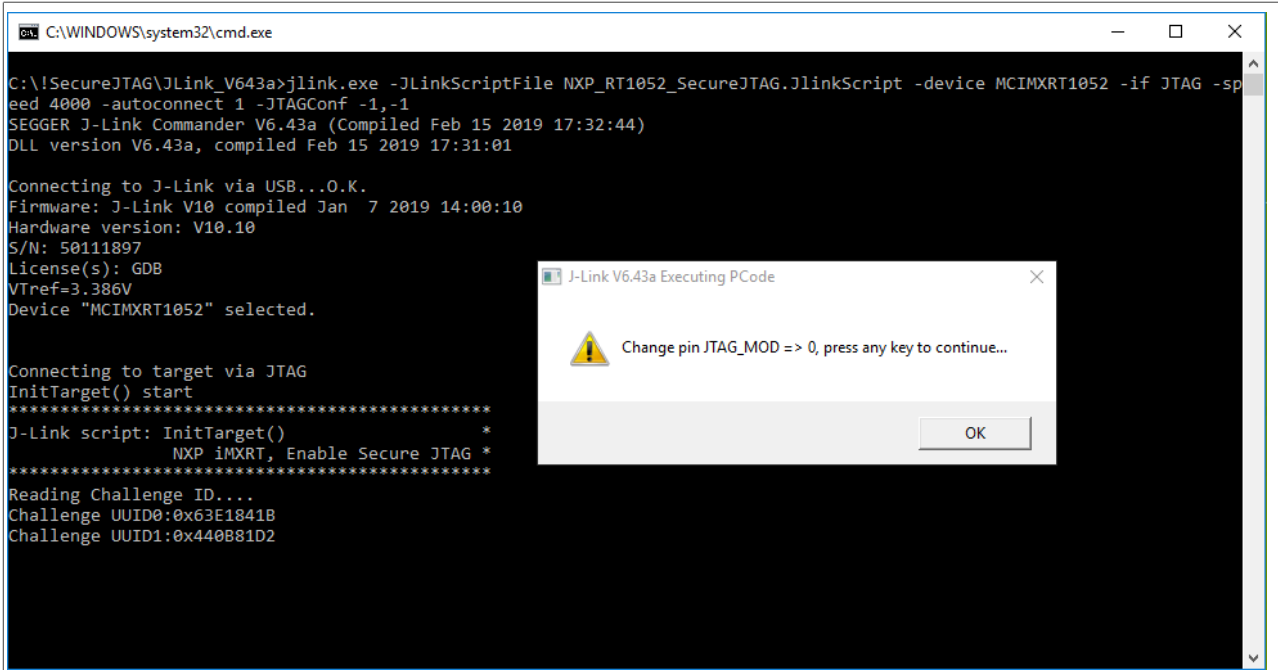


Figure 6. SEGGER J-Link script reads Challenge ID

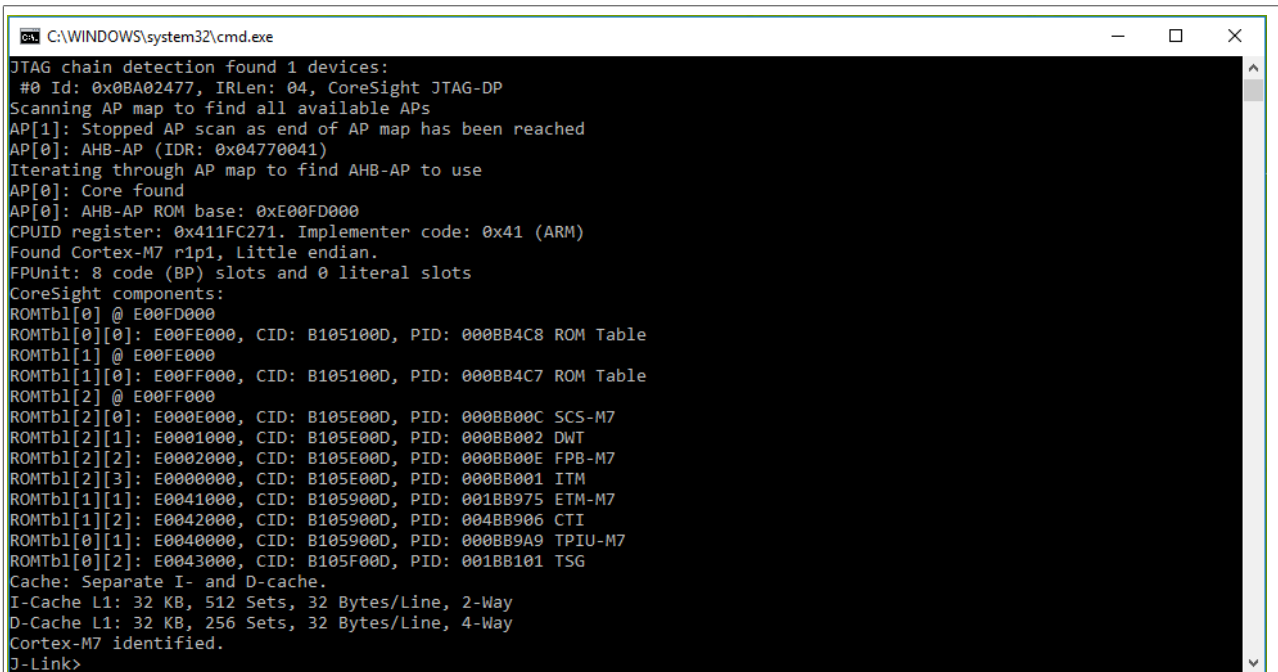


Figure 7. SEGGER J-Link successfully connected to Secured JTAG

Users can now perform normal JTAG debugger operations, as the device has been authenticated using the Challenge-Response mechanism.

Note:

Any reset after JTAG access authorization shifts the JTAG controller back to its lock state, requiring that this authentication process is repeated.

- To ensure that the i.MX RT series SJC is operating in secure mode, edit the "NXP_RT1052_SecureJTAG.JlinkScript" file, provide an incorrect response key, and rerun the script. The debug tool should fail to attach to the i.MX RT10xx series target over JTAG.

4.2 Example of SEGGER J-link Secure JTAG unlock script

```
int InitTarget(void) {
    int v;
    int Key0;
    int Key1;
    JLINK_SYS_MessageBox("Set pin JTAG_MOD => 1 and press any key to
continue...");
    // Secure response stored @ 0x600, 0x610 in eFUSE region (OTP memory)
    Key0 = 0x87654321;
    Key1 = 0xedcba9;
    JLINK_CORESIGHT_Configure("IRPre=0;DRPre=0;IRPost=0;DRPost=0;IRLenDevice=5");
    CPU = CORTEX_M7;
    JLINK_SYS_Sleep(100);
    JLINK_JTAG_WriteIR(0xC); // Output Challenge instruction
    // Readback Challenge, Shift 64 dummy bits on TDI
    JLINK_JTAG_StartDR();
    JLINK_SYS_Report("Reading Challenge ID...");
    // 32-bit dummy write on TDI / read 32 bits on TDO
    JLINK_JTAG_WriteDRCont(0xffffffff, 32);
    v = JLINK_JTAG_GetU32(0);
    JLINK_SYS_Report1("Challenge UUID0:", v);
    JLINK_JTAG_WriteDREnd(0xffffffff, 32);
    v = JLINK_JTAG_GetU32(0);
    JLINK_SYS_Report1("Challenge UUID1:", v);
    JLINK_JTAG_WriteIR(0xD); // Output Response instruction
    JLINK_JTAG_StartDR();
    JLINK_JTAG_WriteDRCont(Key0, 32);
    JLINK_JTAG_WriteDREnd(Key1, 24);
    JLINK_SYS_MessageBox("Change pin JTAG_MOD => 0, press any key to
continue...");
    return 0;
}
```

5 Conclusion

This application note describes the eFuse configuration for Secure JTAG and the authentication process, which is validated and demonstrated using the SEGGER J-Link script.

6 Reference

- Configuring Secure JTAG for the i.MX 6 Series Family of Application Processors* (document [AN4686](#))
- Security reference Manual for the i.MX RT1050 Processor* (document [IMXRT1050SRM](#))
- J-Link / J-Trace User Guide* (document [UM08001](#))
- Training JTAG Interface, Lauterbach TRACE32* (document [Training JTAG Interface](#))
- HAB Code-Signing Tool User's Guide* (Rev. 3.2.0, 04/2019) (document [IMX_CST3.2.0_TOOL](#))

7 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2023 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

8 Revision history

Table 3. Revision history

Document ID	Release date	Description
AN12419 v.3.0	18 December 2023	Section 1 and Section 5 are updated.
AN12419 v.2.0	19 September 2023	The document is updated to correspond to the latest guidelines, Section 1 is updated.
AN12419 v.1.0	11/2019	HAB_JDE bit information is added.
AN12419 v.0.0	04/2019	Initial version

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at <https://www.nxp.com/profile/terms>, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, μ Vision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

i.MX — is a trademark of NXP B.V.

J-Link — is a trademark of SEGGER Microcontroller GmbH.

Contents

1	Introduction	2
2	i.MX RT10xx Secure JTAG support	2
2.1	How to put the chip in Secure JTAG mode	2
2.2	i.MX RT SJC Security Modes	3
2.3	Secure JTAG eFuses	5
2.4	SW Enabled JTAG	5
2.4.1	JDE bit control in HAB (High Assurance Boot)	5
2.5	Secure JTAG debug authentication protocol	6
2.6	SJC disable fuse	7
3	Secret response key approaches	7
3.1	Programming Secure JTAG eFuses using the NXP tool	7
4	Debugging with the Secure JTAG enabled	9
4.1	Steps to connect J-Link debugger via Secure JTAG	9
4.2	Example of SEGGER J-link Secure JTAG unlock script	12
5	Conclusion	12
6	Reference	12
7	Note about the source code in the document	13
8	Revision history	13
	Legal information	14

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.